# The DNS-LU Registry System

Fondation RESTENA

Version 2.0.7

October 18, 2007

# Contents

# Part I

# Introduction

# Chapter 1

# Overview

## 1.1 Introducing registrars

Up to September 2006, an organisation or person wanting to register a ".lu" domain name has to address itself to DNS-LU, i.e. DNS-LU is the only registrar for ".lu" domain names. DNS-LU moreover plays the role of registry for the ".lu" domain. This situation is shown in figure 1.1.

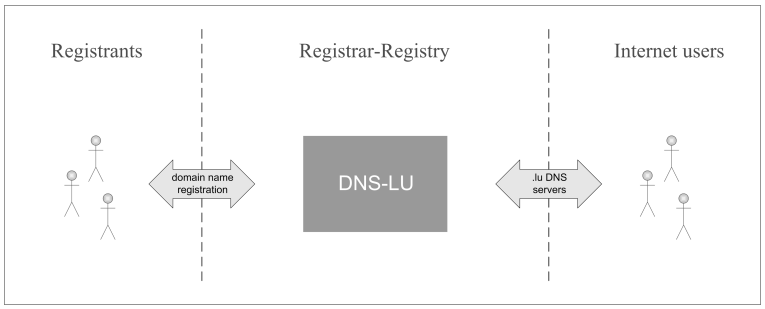


Figure 1.1: Situation before September 2006.

Since the concept of registrars has been adopted, a new situation will arise (fig. 1.2). The interface between the .LU registry and the registrars is heavily based on the Extensible Provisioning Protocol (EPP) with a few minor policy-induced modifications.

Besides the transfer of domain registration data through the EPP protocol, the interface between the registrars and the registry does also include registrar account initialization and the billing of domain name bundles.

The DNS-LU registry acts as a thick registry, i.e. the information about the holder, administrative and technical contacts are maintained at the registry. This is necessary for the Whois service and for the interface between the registry and the registrants (and is very helpful in case of the stop of activities of a registrar).

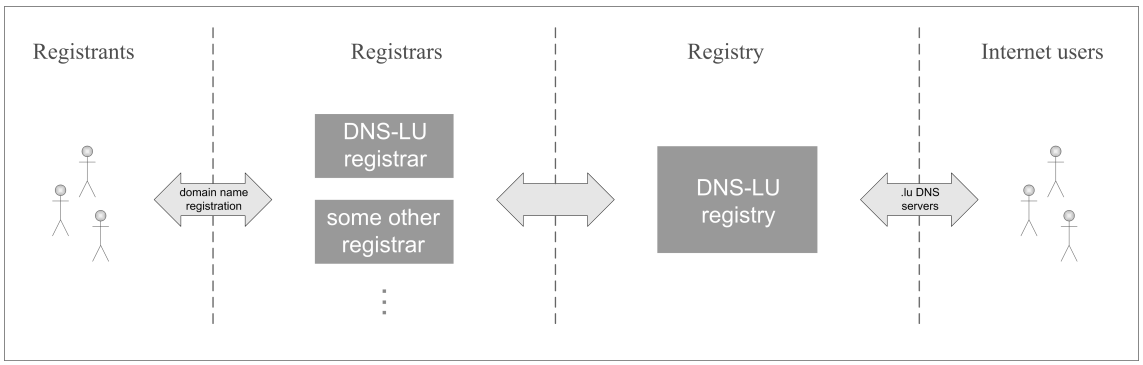


Figure 1.2: Situation "to-be".

# Part II

# Registry policy

# Chapter 2

# General policy elements

## 2.1 Domain name rules

Every domain name that is created at the registry has to conform to the following domain name rules that are defined in the DNS-LU Domain Name Charter:

- It must end in ".lu".
- The part before .lu must be at least 3 characters and at most 63 characters long.
- Before the introduction of IDNs, the domain name must not contain hyphens at the third and fourth position.
- Municipality or villlage names are restricted.
- Allowed characters are a–z, 0–9, and – (hyphen). The hyphen may not be on the first or last position of the domain name.

In general the domain name charter as defined by the group put in place by the government will be the authoritative source for domain names rules.

# Chapter 3

# Object policy and related procedures

## 3.1 Object types and associated data

This section describes how the information, i.e. the data provided by the registrars and stored in the registry database, is structured. Instead of having a datasheet for each domain that contains the related contact and nameserver information (as in the former DNS-LU registrar-registry system), the information is attached to distinct objects, as defined in the EPP protocol. EPP defines three types of objects: domain objects, contact objects and host objects [1]. In order to form the information needed by the DNS, the contact and host objects are dynamically linked to the domain objects. The advantage of dynamically linked objects over static datasheets is that the information corresponding to a particular contact or host is not replicated in each datasheet where it is needed, e.g. a particular host is linked to all the domains for which it acts as a nameserver and the information about the host is present only once in the registry database. This reduces the risk of data inconsistencies, since object updates can be done at a single place.

Every object is owned by exactly one registrar (or, using the terminology of the EPP RFCs, "sponsored by exactly one client"), namely the registrar that created the object or to which an object has been transferred. The owning registrar is the only registrar that can update, delete or retrieve information about an object [2].

In the sequel we will use the term registry-unique to specify objects that have to be unique among all the objects in the registry repository and the term registrar-unique to specify objects that have to be unique only among the objects owned by a particular registrar [3].

In the following sections the different object types are explained in more detail.

### 3.1.1 Domain objects

A domain object corresponds to a domain name (i.e. the name of a second-level subdomain of the .LU ccTLD). For each registered domain name there exists a domain object in the database. Domain objects can be identified by the corresponding domain name and have thus to be registry-unique. The table below shows the information associated to a domain object.

[1] While host objects could refer to any internet hosts, the term *host* is used as a synonym for *name server* in the EPP RFCs and thus also in the context of the DNS-LU registry system.

[2] The only exception to this rule is that any registrar can check whether a particular domain object (identified by the corresponding domain name) exists, i.e. whether the domain name is still available.

[3] Registry-unique objects are also registrar-unique, but the converse is not true.

Table 3.1 lists the information elements associated with domain objects from a registrar's viewpoint, i.e. the information provided by the registrar when he creates or updates a domain object. The registry associates other information with domain objects (such as repository object identifiers, creation date, last update date, etc.) that will be explained in later chapters since those information elements are not provided by the registrars.

| **Domain object** | | | |
|---|---|---|---|
| 1 | **Name** | required | String (6-66, ASCII). |
| 2 | **IDN** [4] | req./opt. | String (6-(variable) [5], UTF-8 encoding). |
| 3 | **Active or reserved** | required | Choice. |
| 4 | **Holder** | required | Reference (i.e. the contact ID) to a contact object containing information about a holder (cf. section 3.1.3). |
| 5 | **Administrative contact** | required | Reference (i.e. the contact ID) to a contact object containing information about a contact (cf. section 3.1.3). |
| 6 | **Technical contact** | required | Reference (i.e. the contact ID) to a contact object containing information about a contact (cf. section 3.1.3). |
| 7 | **Hosts** | req./opt. | At least two references (i.e. the host names) to host objects for an active domain, zero or more references to host objects for a reserved domain (cf. section 3.1.2). |
| 8 | **Status values** | optional | A subset of the status values that are defined in table 3.4. |

Table 3.1: Data associated with domain objects.

**ad 1 :** The domain name has to end in ".lu" and is subject to the rules described in the domain name charter. If the domain name is an IDN, the Name field contains the ACE string. [6]

**ad 2 :** The IDN must be provided if the domain name is an IDN and must not be provided otherwise. If provided, it has to correspond to the ACE string in the Name field.

[4] Not allowed yet.

[5] The decisive factor that determines the length of a domain name is the ACE string for which the maximum length is 66 including the ".lu" suffix. This implies that the maximum length for IDNs is somewhat smaller.

A domain object is always in one and only one of the states listed in the table below.

| **Domain object states** | |
|---|---|
| **state** | **comment** |
| ok | Normal status value. |
| pendingCreate | Period between domain creation command execution and effective creation following visual holder check (village names) and automated nameserver check. |
| pendingUpdate | Between command processing and name server check. |
| pendingDelete | Between command processing and restore or deletion. |

Table 3.2: List of domain object states.

A transfer, trade, transfer-trade or transfer-restore of a domain object passes through different stages. The operation-status, whose values are given in the table below, describes the state of the request.

| **Domain object transfer(-trade)(-restore) / trade status values** | |
|---|---|
| **status value** | **comment** |
| pending | From request submission until changed to one of the other values. |
| clientApproved | Set when registrant (former administrative contact) approves request. |
| clientCancelled | Set when requesting registrar cancels request. Remains until the request is deleted (history purpose). |
| clientRejected | Set when registrant (former administrative contact) rejects request. Remains until the request is deleted (history purpose). |
| serverCancelled | Set when registry aborts operation (timeout). Remains until the request is deleted (history purpose). |

Table 3.3: List of domain object transfer, transfer-restore, transfer-trade and trade status values.

The status values that can be attached to a domain object are listed in the table below. As specified in the EPP RFCs, status values that are prefixed with "client" can be added or removed by the registrar while status values that are prefixed with "server" can be added or removed only by the registry.

| **Domain object status values** | | |
|---|---|---|
| **status** | **by** [7] | **comment** |
| serverUpdateProhibited | R | Requests to update the object are rejected. |
| serverDeleteProhibited | R | Requests to delete the object are rejected. |
| serverRestoreProhibited | R,E | Requests to restore the object are rejected. |
| serverTradeProhibited | R,E | Requests to trade or transfer-trade the object are rejected. |
| serverTransferProhibited | R | Requests to transfer, transfer-trade or transfer-restore the object are rejected. |
| serverHold | R | DNS delegation information is not exported into the zone data file. |
| clientHold | C | |
| serverRenewProhibited | R | Prevent automatic renewal of the object. |
| clientRenewProhibited | C | |
| inactive | C | Reserved domain |

Table 3.4: List of domain object status values.

### 3.1.2 Host objects

Host objects have to be registrar-unique (i.e. a registrar cannot own two host objects that are both named ns.domain.lu) but not registry-unique (i.e. registrar1 can own a host named ns.domain.lu and registrar2 can own a host named ns.domain.lu) [8].

| **Host** | | | |
|---|---|---|---|
| 1 | **Name** | required | String (1-255, ASCII [9]) |
| 2 | **IPv4 address** | optional | String (15, ASCII) [10] |
| 3 | **IPv6 address** | optional | String (3-39, ASCII) [11] |

Table 3.5: Data associated with host objects.

[7] Flags to give extra information on status item usage. E = DNS-LU extension, C = Can be set by registrar, R = Can by set by registry only. If none is given, then the EPP status flag is NOT used

[8] As for contact objects, this is motivated by the fact that if host objects would be registry-unique (and thus be used by several registrars) updates on the object by the owner might imply an inconsistency with the data in the database of another registrar using the object.

[9] If the host name includes an IDN, the IDN has to be given in its ACE encoding.

[10] Sanity syntax checks will be performed on IPv4 addresses.

[11] Sanity syntax checks will be performed on IPv6 addresses.

**ad 2, 3 :** The IPv4 and IPv6 addresses are optional, but if a glue record is necessary at least one address has to be provided since otherwise an error occurs during the creation of the corresponding domain. It should be noted that at most one IPv4 and at most one IPv6 address can be provided.

| **Host object states** | |
|---|---|
| ok | Normal status value. |
| pendingUpdate | Between command processing and name server check. |

Table 3.6: List of host object states.

The status flags that can be set on host objects are listed below. The linked status is calculated.

| **Host object status values** | |
|---|---|
| serverUpdateProhibited | Requests to update the object are rejected. |
| serverDeleteProhibited | Requests to delete the object are rejected. |
| linked | The contact object is linked to at least one domain object. |

Table 3.7: List of host object status values.

### 3.1.3 Contact objects

Contact objects are used to hold information about both domain holders and contacts, i.e. a contact object is either a contact object holding information about a domain name holder or a contact object holding information about a contact (administrative or technical) [12]. The same contact can be used as both administrative and technical contact of a domain, i.e. the contact (as a contact object in the registry database) does not hold information whether it is an administrative or a technical contact. The distinction between administrative and technical contacts is made during domain creation, when contacts can be linked to the domain as an administrative and/or a technical contact.

Contact objects have to be only registrar-unique.

| **Contact object (holder)** | | | |
|---|---|---|---|
| 1 | **Contact ID** | required | String (3-16, ASCII). |
| 2 | **Organisation or private person** | required | Choice. |
| 3 | **Name** | required | String (1-255, ASCII + latin-1). |

[12] In this document, we refer to a *contact object* as the object type defined in EPP and that can hold either information about a holder or a contact, and we refer to a *contact* as the contacts to be used as both administrative or technical contacts for a domain, as opposed to holders.

| | | | |
|---|---|---|---|
| 4 | **Address 1** | required | String (1-255, ASCII + latin-1). |
| 5 | **Address 2** | optional | String (0-255, ASCII + latin-1). |
| 6 | **Address 3** | optional | String (0-255, ASCII + latin-1). |
| 7 | **Postal code** | required | String (1-16, ASCII). |
| 8 | **City** | required | String (1-255, ASCII + latin-1). |
| 9 | **Country code** | required | String (2, ISO 3166-1). |
| 10 | **Disclosure preferences** | optional | A subset of the following: name, address. |

Table 3.8: Data associated with contact objects (holder).

**ad 4, 5, 6 :** The first of the three address fields will generally be sufficient to enter the street address. The additional address fields can be used if other elements such as a department, a state or a province has to be specified.

**ad 10 :** Cf. **ad 13** table 3.9.

| **Contact object (contact)** | | | |
|---|---|---|---|
| 1 | **Contact ID** | required | String (3-16, ASCII). |
| 2 | **Name** | required | String (1-255, ASCII + latin-1). |
| 3 | **Organisation** | optional | String (0-255, ASCII + latin-1). |
| 4 | **Address 1** | required | String (0-255, ASCII + latin-1). |
| 5 | **Address 2** | optional | String (0-255, ASCII + latin-1). |
| 6 | **Address 3** | optional | String (0-255, ASCII + latin-1). |
| 7 | **Postal code** | required | String (1-16, ASCII). |
| 8 | **City** | required | String (1-255, ASCII + latin-1). |
| 9 | **Country code** | required | String (2, ISO 3166-1). |
| 10 | **Telephone number** | optional | String (0-17, ASCII) [13]. |
| 11 | **Fax number** | optional | String (0-17, ASCII). |
| 12 | **E-mail address** | required | String (1-255 [14], ASCII [15]) |

[13] As defined in RFC3733: Telephone numbers are character strings that MUST begin with a plus sign ("+", ASCII value 0x002B), followed by a country code, followed by a dot (".", ASCII value 0x002E), followed by a sequence of digits representing the telephone number.

[14] cf. RFC 2821.

[15] If the domain part of the e-mail address includes an IDN, the IDN has to be given in its ACE encoding.

| | | | |
|---|---|---|---|
| 13 | **Disclosure preferences** | optional | A subset of the elements name, organisation, address, telephone number, fax number, e-mail address, and a specification for each element whether to allow or restrict disclosure. |

Table 3.9: Data associated with contact objects (contact).

**ad 13 :** The registrar can modify the default disclosure policy for allowing or restricting disclosure by the Whois service by specifying a subset of the elements listed in the table, e.g. if the default disclosure policy stipulates that the telephone number is disclosed, the registrar can specify that for a particular contact object the telephone number has to be undisclosed, or vice-versa. The address element includes the three address fields, the postal code, the city and the country code.

The status flags that can be set on contact objects are listed below. The linked status is calculated.

| **Contact object status values** | |
|---|---|
| serverUpdateProhibited | Requests to update the object are rejected. |
| serverDeleteProhibited | Requests to delete the object are rejected. |
| linked | The contact object is linked to at least one domain object. |

Table 3.10: List of contact object status values.

## 3.2 Object manipulation procedures

In order to transfer the data on domains, contacts and name servers to the registry and to subsequently manage that data, the registrars issue requests to create, update, trade, transfer, transfer-trade, transfer-restore, restore or delete an object towards the registry. The registrars have moreover the possibility to check the availability of an object or retrieve information about an object [16]. The registrars have to provide the information that the registry system needs to execute the requests. The sections below roughly describe the procedures associated with each of the requests. The procedures will be given in explicit detail in later chapters. One of the important points is the distinction between requests that can be executed immediately by the registry system and requests that need some subsequent processing. For requests that can be executed immediately, the requesting registrar is instantaneously informed about the success or failure of the request execution. For requests that need some subsequent processing, i.e. that will be pending, the requesting registrar is instantaneously informed that the request is registered and gets information about the success or failure of the request exection only after the postponed processing. In this chapter we thus limit our description of the registry system to the following:

[16] The request types listed in this chapter correspond to the commands defined by the EPP protocol that will be described in chapter 5. The object manipulation policy described in this section is derivated from the EPP protocol.

it parses the request, executes the request immediately if possible, otherwise registers the request and queues it for later processing.

### 3.2.1 Object creation

A registrar can issue an object creation request in order to create an object in its repository view.

#### 3.2.1.1 Domain object creation

A domain object can be created if there does not already exist [17] a domain object with the same domain name in the registry repository [18] and if the domain name is not in the list of domain names that are rejected by the registry due to some special reason. Furthermore, the domain name has to respect the domain name rules as laid out in the domain name charter. The contact and host objects that are referenced have to exist in the registrar's repository view [19]. If the domain object is requested to be active, at least two host references have to be provided. If a glue record [20] is necessary, the IP address should be provided for the concerned host objects since otherwise the subsequent name server test will fail.

After parsing the request, the registry system registers the domain object creation request (indeed it registers a domain object with state pendingCreate) that is queued. If the visual check succeeds (holder check for village names or auto-success for other valid domain names) and if the domain object is requested to be active, a name server test is performed. Otherwise, if the visual check succeeds but the domain is requested to be reserved the domain object is created. If, for an active domain object, the name server test succeeds, the domain object is created as an active domain object. Otherwise, if the name server test fails, the domain object is created as a reserved domain object [21], i.e. an activation request using the domain update procedure has to be issued afterwards in order to activate the domain.

It is important to note that a domain object is considered to be created at the moment when it is not in state pendingCreate anymore, i.e. not at the moment when the domain creation request is registered [22]. The creation date of the domain object is thus set to that date and the domain will expire one year after that date (and not one year after the date when the creation request has been registered) [23].

If the name of the domain object to be created is a village name, a human check is made to validate that the holder corresponds to the local administration that is authorized to register the

[17] More precisely, a domain object can be created if there does not already exist a domain object with the same domain name in the registry repository (even if that domain object is in quarantine, i.e. in state pendingDelete) and if no creation request for a domain object with the same domain name has already been issued by some registrar (i.e. there must not be a domain object with the same domain name that is in state pendingCreate).

[18] Remember that, since domain objects are registry-unique, a domain object cannot be created it there already exists a domain object with the same domain name in the whole registry repository, and not only in the registrar's repository view.

[19] The registrar's repository view is the set of objects in the registry repository owned by the registrar.

[20] A glue record, i.e. an A record that provides the IP address of a name server used in an NS record for delegation purposes, is necessary if a name server is used as a name server for its parent domain (e.g. if `ns.domain.lu` is used as a name server for `domain.lu`). In the context of the DNS-LU registry, glue records may be required only for local name servers, i.e. name servers which names end in ".lu", since the delegation is only made for domains ending in ".lu". It should be noted that a glue record is not necessary for every local name server since a situation might exist where a local name server is not used as a name server for its parent domain, i.e. other name servers are used for the delegation of the parent domain.

[21] It is important that even if the activation fails the domain name remains reserved, i.e. the corresponding domain object cannot be created by another registrar.

[22] The date when the domain creation request is registered is the date from which on the domain name is reserved.

[23] This means that one does not have to pay for the domain during the period when it cannot yet be actively used.

domain name [24].

#### 3.2.1.2 Host object creation

A host object can be created if there does not already exist a host object with the same host name in the registrar's repository view, without further constraints [25].

The request for a host object creation is immediately executed by the registry system. IP addresses must not be provided for non-local host objects, i.e. host objects whose name do not end in ".lu". A local host object can be created with or without IP addresses, but if a glue record will be necessary, i.e. if the host will be used as a name server for its parent domain, at least one IP address should be provided since otherwise the activation of the parent domain will fail [26].

#### 3.2.1.3 Contact object creation

A contact object can be created if there does not already exist a contact object with the same contact ID in the registrar's repository view. The request for a contact object creation is immediately executed by the registry system. In order to take advantage of the concept of objects, a contact object should only be created if there is not already a contact object for the same physical person or organisation in the registrar's repository view. If the information attached to a contact object have to be modified, the corresponding contact object should be updated instead of creating a new one.

### 3.2.2 Object update

A registrar can issue an object update request in order to update an object that exists in his repository view.

#### 3.2.2.1 Domain object update

A domain object can be updated if it is neither pendingUpdate, pendingCreate or pendingDelete. The serverUpdateProhibited status flags also prevents updates. If a part of an update requires pending update, then the whole command will be pending and succeeds or fails atomically. The following updates can be performed on a domain object [27]:

---

[24] It should be noted that domain objects that correspond to a village name can in principle not be traded. The registry system therefore sets the status value serverTradeProhibited for those domain objects when they are created. Setting this status value for all domain objects that correspond to a village name ensures that such a domain object cannot be traded. If it occurs that a village name has to be traded, the registrant or the registrar has to contact the registry, and the registry administrator has to manually remove that status value until the domain object is traded.

[25] While the parent domain of a local name server, i.e. a name server which name ends in ".lu", has to exist in the zone data file, this is not true for the registry repository. A local host object can be created even when the parent domain object does not yet exist. This is motivated by the fact that otherwise if a registrant would want to use some name servers for the delegation of their parent domain, the registrar would have to first create the domain object as a reserved domain object, then create the host objects and afterwards update the domain object in order to link the host objects to the domain object and activate the domain object. Allowing local host objects to be created when the parent domain does not exist makes it possible for the registrar to first create the host objects and then link them with the parent domain when it is created.

[26] It is not possible to determine whether a glue record is necessary or to perform a name server test when a host object is created since at that moment the host object cannot already be linked to some domain object. It is moreover not possible to check whether a provided IP address does correspond to the provided host name since this would imply that the parent domain already exists in the zone data file what is against the principle that (local) host objects can be created without the existence of the parent domain (a test whether an IP address corresponds to the host name should anyway only be done if the IP address will indeed be used). A check whether a glue record is necessary and a name server test will be performed when the host object will be linked to a domain.

[27] It should be noted that, even if not listed in this section, the reference to the holder of a domain object can also be updated. Since that update requires some special processing it is not considered as a simple domain object

- Add or remove status values: The status values "inactive" and "clientHold" can be added (and others if they are in use by the registry) or removed (if they are associated to the domain object). If the update command does only consist of additions and/or removals of status values, the update is executed immediately. If the update command contains update requests that require the update to be pending, the status values are updated together with the other elements.

  A status value may not be added and removed in the same command.

- Switch from reserved to active: Causes an update to be pending. In order to switch a domain object from reserved to active, links to at least two different host objects have to already exist or they have to be provided in the same request. The update is pending (the state of the domain object is switched to pendingUpdate) until a subsequent name server test is performed. While two hosts are enough to activate a domain, all the hosts linked with the domain have to work and will thus be tested during the name server test. If the name server test succeeds the domain object is switched from reserved to active. Otherwise the command fails and the domain remains reserved, i.e. in order to activate the domain a new activation request for the domain object has to be issued afterwards. Activation of the domain is done be removing "inactive" status flag.

- Switch from active to reserved: This update is executed immediately, even if hosts are added or removed in the same command. In general, if the domain will be reserved after successful command execution, a subsequent name server test is not necessary. The domain is switched to reserved by adding the flag "inactive".

- Change the administrative or technical contact: The administrative and/or technical contacts associated with a domain can be changed by providing new contact references. If the update command does only consist of the change of the contacts, the update is executed immediately. Please note that the administrative or technical contact of a domain should be updated if the physical person that acts as contact is changed. If on the other hand some information elements of the physical person change, then the corresponding contact object should be updated instead of creating a new object and update the link to the domain object.

- Add a host: New references to host objects can be added if the number of hosts linked to this domain will not exceed 10. Several hosts can be added or removed in the same request. The number of remaining hosts after request execution, i.e. taking into account also the hosts to be removed, must not exceed 10. If the domain is active or if an activation request is issued in the same request, the domain object is switched to pendingUpdate until a subsequent name server test is performed. If the name server test for succeeds and if the name server tests required for some other update requests in the same request do also succeed [28], the action is executed. If the name server test fails the registrar will have to make a new update request afterwards in order to add the hosts. If the domain is reserved or if a de-activation request, i.e. a switch from active to reserved, is issued in the same request, the action is executed immediately.

- Remove a host: References to host objects can be removed if the domain is reserved or, for an active domain, if the number of hosts linked to this domain will not be lower than 2 [29].

update, but as a domain object trade, as defined in section 3.2.5. It should moreover be noted that the name of a domain object cannot be updated (for every domain name a domain object has to be created).

[28] A registrar's request has to succeed or fail as a whole.

[29] Several hosts can be removed or added in the same request. The number of remaining hosts after command execution, i.e. taking into account also the hosts to be added, must not be lower than 2.

If the domain is active or if an activation request is issued in the same request, the domain object is switched to pendingUpdate until a subsequent name server test [30] is performed. If the name server test succeeds and if the name server tests required for some other update requests in the same request do also succeed [31], the action is executed. If the name server test fails the registrar will have to make a new update request afterwards in order to remove the hosts. If the domain is reserved or if a de-activation request is issued in the same request, the action is executed immediately.

#### 3.2.2.2 Host object update

A host object can be updated if it has state ok (i.e. if it is not already in state pendingUpdate) and does not have serverUpdateProhibited as associated status value. The following updates can be performed on a host object:

- Name update: The name of a host can be updated, but this action has to be treated with care [32]. If the current host name is local, the new host name must also be local, and if the current host name is non-local, the new host name must also be non-local, since the address policy differs for local and non-local hosts. If an IP address update that is not executed immediately (i.e. that requires state pendingUpdate) is inquired in the same request, the action will be pending [33] and the state of the host object will be set to pendingUpdate. In that case the action is executed if the IP address update succeeds and is ignored if the IP address update fails. Otherwise, if no such pending IP address update is inquired in the same request, the action is executed immediately.
- Add an IPv4 or an IPv6 address: An IPv4 (respectively IPv6) address can be added if the IPv4 (respectively IPv6) address is not already set for the host object (otherwise the previous address must be removed in a previous or in the same command) and if the host object is local, i.e. if its name ends in ".lu". If the parent domain is linked to the host and is active (in that case a glue record is necessary), the action will be pending (the state of the host object will be switched to pendingUpdate until a subsequent name server test is performed [34]. If the name server test succeeds the IPv4 (respectively IPv6) address is added to the host object, otherwise the address is not added. If the parent domain is reserved [35] or not linked to the host (or does not yet exist in the registrar's repository view) and if there is no pending IPv6 address (respectively IPv4 address) removal or change requested in the same command, the action is executed immediately (In that case there is no need to perform a name server test

[30] The name server test for the removal of hosts tests whether the remaining hosts are correctly configured. It seems careful that possibly correctly configured hosts cannot be removed if the remaining hosts are not correctly configured.

[31] Remember that a request has to succeed or fail as a whole. It seems moreover careful to not remove possibly correctly configured hosts if the newly added hosts are not correctly configured. This implies that in a situation where an active domain uses 10 incorrectly configured hosts, a request for the removal of the 10 hosts and the addition of at least 2 new hosts has to be issued.

[32] The name of a host should only be updated when the name of the server is changed, i.e. the name server has to keep the same configuration. The name of a host should not be updated if a host of a linked domain is updated because this name change affects ALL domains that link to this host. In that case a new link should be created.

[33] If updates that are not executed immediately are inquired in the same request, the name change has to be made pending too since otherwise the principle that a command can succeed or fail only as a whole might be broken.

[34] It should be noted that even if the parent domain will be involved in the subsequent name server test, the parent domain object is not made pending. This corresponds to the principle that a domain object can not be made pending if a linked object is updated, i.e. if no action is performed on the domain object itself.

[35] It is allowed that a reserved domain object has host objects associated to it. This makes it possible that host objects can be linked to a reserved domain that may be activated at a later moment. Moreover this makes it easier to de-activate a domain and re-activate at a later moment with the same name servers.

since a glue record is not needed. A name server test using the IP address will be performed when the host object is linked to its parent domain.) [36].

- Remove the IPv4 or the IPv6 address: When the parent domain is linked to the host object and is active, the IPv4 (respectively IPv6) can only be removed if an IPv6 (respectively IPv4) address exists for the host or is requested to be added in the same request (since when a glue record is necessary at least one address (IPv4 or IPv6) has to be provided). If the host is not linked to the parent domain (or if the parent domain is reserved or does not exist in the registrar's repository view) and if there is no pending IPv6 address (respectively IPv4 address) addition in the same command, the action is executed immediately. Otherwise the host object is set to pendingUpdate until a subsequent name server test will be performed in order to test whether the remaining IP address works.

#### 3.2.2.3 Contact object update

A contact object can be updated if it does not have the serverUpdateProhibited status value set. Name and address elements as well as disclosure preferences can be updated, and the update request is executed immediately. However, updates to the name and address fields are only allowed as long as the legal entity that is described remains the same. If the legal holder of the domain name changes, a trade or transfer-trade has to be requested [37].

### 3.2.3 Object deletion

A registrar can issue an object delete request in order to delete an object that exists in his repository view.

#### 3.2.3.1 Domain object deletion

A domain object can be requested to be deleted if it is not in any pending state [38] [39] and does not have serverDeleteProhibited as associated status value. A deletion date and time, that specifies the day and time when the deletion process shall be started [40], may be specified (if a registrar requests the deletion of a domain object with specification of a deletion date, the domain object may be in state pendingUpdate (but not in state pendingCreate or pendingDelete) when that request is issued [41], but the status value serverDeleteProhibited must not be set when the request is issued). If the deletion date is not specified by the registrar, the deletion process is started immediately. The deletion process involves that the domain object is switched to state pendingDelete during

[36] IP addresses can be added to a host object even if no glue record is necessary since host objects can also be created with IP addresses if no glue record is necessary, as described in section 3.2.1.2.

[37] As a result, the field "organisation or private person" of a holder cannot be updated.

[38] A domain object can thus not be requested to be deleted if it is in state pendingCreate, pendingUpdate or already in state pendingDelete. Since a domain object may be in state pendingUpdate only during a fairly short period, a registrar can wait until the domain object is again in state ok if the wants to delete a domain object that is in state pendingUpdate.

[39] A domain object can thus be requested to be deleted if a trade, transfer, transfer-trade has been requested for the domain object. For more information on those situations we refer to the corresponding sections.

[40] The deletion date that may be provided by the registrar specifies the date when the domain object should be set to state pendingDelete and not the date when the domain object is indeed deleted (after the quarantine period).

[41] The domain object may be in state pendingUpdate when a deletion with specification of a deletion date is requested, but it must not be in state pendingUpdate at the deletion date when the domain object should be set to pendingDelete. This does not cause a problem since a domain object may be in state pendingUpdate only during a fairly short period, and the domain object will be set to pendingDelete by the registry system when it is again in state ok.

some quarantine period [42]. Active domain objects are not exported any more to the DNS-LU servers when they enter the quarantine period. During the quarantine period, the holder of the domain has the possibility to restore the domain, i.e. to undo the deletion (cf. 3.2.4.1). If no restore is requested by the holder during the quarantine period, the domain object is deleted and the domain name is free for a new registration.

#### 3.2.3.2 Host object deletion

A host object can be deleted if it is not in any pending state and does not have serverDeleteProhibited or linked as associated status value. The deletion is executed immediately.

It should be noted that a host object that is not linked to a domain object during an extended period is automatically deleted by the registry.

#### 3.2.3.3 Contact object deletion

A contact object can be deleted if it is not in any pending state and does not have serverDeleteProhibited or linked as associated status value. The deletion is executed immediately.

It should be noted that a contact object that is not linked to a domain object during an extended period is automatically deleted by the registry.

### 3.2.4 Object restore

An object restore request does only apply to domain objects. A registrar can issue a domain object restore request in order to restore a domain object that exists in his repository view and that is in quarantine (i.e. in state pendingDelete).

#### 3.2.4.1 Domain object restore

A domain object can be restored if it is in state pendingDelete and does not have serverRestoreProhibited as associated status value. Furthermore, a domain object can only be restored if no transfer-restore has already been requested for the domain object. The domain object will be restored with the same attributes and links that it had when it was requested to be deleted. The action is executed immediately.

### 3.2.5 Object trade request

An object trade request does only apply to domain objects. A registrar can issue a domain object trade request in order to trade a domain object, i.e. in order to update the reference to the holder of a domain object [43], that exists in his repository view.

[42] It should be noted that a domain object is in state pendingDelete when it is in the quarantine period and vice-versa (both expressions can be interchangeably used to denote the same thing).

[43] The update of the reference to the holder of a domain object corresponds to the update of a property of the domain object and could therefore be considered as a part of a domain update request. But since this update constitutes a major issue from an administrative viewpoint and should not be confused with some other domain update and since it requires some special procedure, it has been decided to consider the update of the holder reference separately from the other domain updates.

#### 3.2.5.1 Domain object trade request

A domain object can be traded if it does not have a serverTradeProhibited [44] status value. Furthermore, a domain object can only be traded if no transfer, transfer-trade or another trade has already been requested for the domain object. It should be noted that a domain object can be traded even if it is in state pendingUpdate [45]. Moreover, there is no explicit pendingTrade status value that would be set while the registry system processes the trade of the domain object [46].

The trade of a domain object can only be requested by the registrar that owns the domain object. A trade request is similar to a create request, since all the attributes of the domain object have to be provided in the request.

When a trade request is issued, the registry system registers the trade request and sets the status of the trade request to *pending* until some subsequent processing is performed. The subsequent processing starts with sending an approval request by email to the former (indeed the current) administrative contact, i.e. the administrative contact currently linked with the domain object. The former administrative contact has to approve or reject the trade either by using a special webpage, or an off-line communication (e.g. faxsimile). If the trade is rejected, the trade is cancelled by the registry system and the trade-status is set to clientRejected. Else, if the trade is approved, the trade-status is set to clientApproved. If the domain is requested to be active a name server test is performed. If the name server test fails, the domain will be set to reserved. Whether the name server test fails or not, the trade is executed. The registrar is informed (by a poll-message) that the trade has been executed and whether the domain has been switched from active to reserved due to a name server test failure. If the domain was active before the trade, an email that states that the former name servers should possibly not be authoritative for the domain anymore is sent to the former technical contact (If the new name servers are different from the former name servers, the registrar or the former technical contact (depending on who manages the old name servers) has to update the configuration of his name servers. Note that if the name servers remain the same, there is no need to update the name servers.).

The former administrative contact has a predefined period to approve or reject the trade. After the expiration of half that period an information email is sent to the registrar to allow him to contact his customer. If after the defined period no approval or rejection has been received, the trade request is canceled. The old administrative contact cannot approve or reject the trade by means of the registrant web interface when the trade has already been cancelled (due to period exceeding or explicit cancellation (see section 3.2.7)).

The requesting registrar can optionally specify a date and time at which the trade should be executed (we will refer to this date as the trade-date). In that case, the processing described above is performed directly when the request is issued but only up to the approval. The name server test and the trade execution are performed at (or shortly after) the specified date and time.

[44] The status value serverTradeProhibited is automatically set by the registry system for domain objects that correspond to a village name at the moment when they are created. If a domain object that corresponds to a village name has to be traded, the registry has to manually remove that status value from the domain object.

[45] It should be noted that a domain object cannot be traded if it is in state pendingCreate or pendingDelete. If one wants to trade a domain object that is in state pendingCreate one has to wait for the domain object to be created by the registry system before trading it. If one wants to trade a domain object that is in state pendingDelete, one has to first restore the domain object and then trade it or wait until the domain object will be deleted after the quarantine period and newly register the domain afterwards (with the risk that someone other is faster in registering the domain).

[46] It should be noted that a domain object can be deleted while a requested trade of the domain object has not yet been executed. In that case, the trade is cancelled by the registry system and the registrar is notified about that cancellation. Moreover, if a trade approval request has already been sent to the former administrative contact, this one is also informed about the cancellation of the trade.

### 3.2.6 Object trade query

An object trade query does only apply to domain objects. A registrar can issue a domain object trade query in order to get information about a pending domain object trade, i.e. an ongoing domain object trade that has not yet been executed by the registry, that he previously requested by means of a domain object trade request. A domain object trade query cannot be issued for a domain object trade request that has already been executed by the registry.

#### 3.2.6.1 Domain object trade query

As already mentioned, a domain object trade query can be issued for a previously requested domain object trade that has not yet been executed by the registry. Only the registrar that owns the domain to be traded, i.e. the registrar that requested the domain object trade, can issue a domain object trade query. The information provided to the registrar in response to the domain object trade query is essentially the trade-status. The trade-status shows the current stage of the ongoing trade procedure as described in section 3.2.5.1.

### 3.2.7 Object trade cancellation

An object trade cancellation does only apply to domain objects. A registrar can issue a domain object trade cancellation in order to cancel a pending domain object trade, i.e. an ongoing domain object trade that has not yet been executed by the registry, that he previously requested by means of a domain object trade request. A domain object trade cancellation cannot be issued for a domain object trade request that has already been executed by the registry.

#### 3.2.7.1 Domain object trade cancellation

As already mentioned, a domain object trade cancellation can be issued for a previously requested domain object trade that has not yet been executed by the registry. Only the registrar that owns the domain to be traded, i.e. the registrar that requested the domain object trade, can issue a domain object trade cancellation. If a trade approval request has already been sent to the former administrative contact, the registry informs him by email about the domain object trade cancellation. [47]

### 3.2.8 Object transfer request

An object transfer request does only apply to domain objects. A registrar can issue a domain object transfer request in order to transfer a domain object that does not exist in his repository view from the current registrar's repository view into his own repository view, i.e. the term transfer denotes the change of the registrar that owns the domain object.

#### 3.2.8.1 Domain object transfer request

The domain object transfer request procedure is similar to the domain object trade request procedure (cf. section 3.2.5.1), except that there are two registrars involved.

A domain object can be transferred if it does not have a serverTransferProhibited status value. Furthermore, a domain object can only be transferred if no trade, transfer-trade or another transfer has already been requested for the domain object. As for a domain trade, a domain object can be

[47] Please note that the registry does not have to re-increment the registrar's credit because of the cancellation, since the registrar's credit has not been decremented so far. The registrar's credit would have been decremented at the moment when the registry would have executed the domain object trade request.

transferred even if it is in state pendingUpdate [48] and there is no explicit pendingTransfer status value [49].

Unlike a domain trade, the transfer of a domain object can only be requested by the new registrar, i.e. the registrar that will become the owner of the domain object. The registrar that owns the domain object must not request a transfer. As for a domain trade, a transfer request is similar to a create request, since all the attributes of the domain object have to be provided in the request (if they do not already exist, the host and contact objects linked to the domain have to be newly created in the new registrar's repository view).

As for a trade request, when a transfer request is issued, the registry system registers the transfer request and sets the status of the transfer request to *pending* until some subsequent processing is performed.

At this point, the procedure is the same as the procedure in case of a domain trade request, i.e. an approval request is sent by email to the former administrative contact and so on.

Unlike a domain trade, when the transfer is executed, two registrars (the new, who requested the transfer, and the former), instead of only one, are informed about the transfer execution.

As for a domain trade, a transfer-date may be specified. The processing in that case is the same as for a domain trade.

### 3.2.9 Contact object copy on transfer request

When requesting a domain transfer the contacts can be requested to be copied from current registrar to the requesting registrar. For privacy reasons that copied data is NOT available to the registrar until the domain transfer has been successfully executed.

The copy of data happens at the time of domain transfer request and the newly created contact object is tagged as copy. This tag gets removed on successful transfer of the domain it is attached to. As long as the tag exists the contact details cannot be querried and the contact cannot be used for any purpose. If the domain transfer fails the contact can be manually deleted before automatic expiration of unused contacts.

When copy on transfer is used for registrants it is not possible to merge multiple identical contact objects to have all domains of the contact associated to a single contact object unless a trade is done.

### 3.2.10 Object transfer query

An object transfer query does only apply to domain objects. A registrar can issue a domain object transfer query in order to get information about a pending domain object transfer that he previously requested by means of a domain object transfer request. A domain object transfer query cannot be issued for a domain object transfer request that has already been executed by the registry.

#### 3.2.10.1 Domain object transfer query

The domain object transfer query procedure is the same as the domain object trade query procedure (cf. section 3.2.6.1). Note that only the registrar that issued the domain object transfer request, i.e. the"new" registrar, can issue a domain object transfer query. The registrar that

[48] If one wants to transfer a domain object that is in state pendingDelete, one has to use the domain object transfer-restore request as described in section 3.2.15.

[49] It should be noted that a domain object can be deleted while a requested transfer of the domain object has not yet been executed. In that case, the transfer is cancelled by the registry system and the new registrar that requested the transfer is notified about that cancellation. Moreover, if a transfer approval request has already been sent to the former administrative contact, this one is also informed about the cancellation of the transfer.

currently owns the domain object, i.e. the "former" registrar, cannot get information about the ongoing domain object transfer request.

### 3.2.11 Object transfer cancellation

An object transfer cancellation does only apply to domain objects. A registrar can issue a domain object transfer cancellation in order to cancel a pending domain object transfer that he previously requested by means of a domain object transfer request. A domain object transfer cancellation cannot be issued for a domain object transfer request that has already been executed by the registry.

#### 3.2.11.1 Domain object transfer cancellation

The domain object transfer cancellation procedure is the same as the domain object trade cancellation procedure (cf. section 3.2.7.1). Note that only the registrar that issued the domain object transfer request, i.e. the "new" registrar, can issue a domain object transfer cancellation. The registrar that currently owns the domain object, i.e. the "former" registrar, cannot cancel the ongoing domain object transfer request.

### 3.2.12 Object transfer-trade request

An object transfer-trade request does only apply to domain objects. A registrar can issue a domain object transfer-trade request in order to simultaneously transfer and trade a domain object. [50]

#### 3.2.12.1 Domain object transfer-trade request

The domain object transfer-trade request procedure is the same as the domain object transfer request procedure (cf. section 3.2.8.1). A domain object transfer-trade request can only be issued by a registrar that is not currently the owner of the domain object.

A domain object can be transfer-traded if it does not have any serverTransferProhibited or serverTradeProhibited status value. Furthermore, a domain object can only be transfer-traded if no trade, transfer or another transfer-trade has already been requested for the domain object.

### 3.2.13 Object transfer-trade query

An object transfer-trade query does only apply to domain objects. A registrar can issue a domain object transfer-trade query in order to get information about a pending domain object transfer-trade that he previously requested by means of a domain object transfer-trade request. A domain object transfer-trade query cannot be issued for a domain object transfer-trade request that has already been executed by the registry.

#### 3.2.13.1 Domain object transfer-trade query

The domain object transfer-trade query procedure is the same as the domain object trade query procedure (cf. section 3.2.6.1). Note that only the registrar that issued the domain object transfer-trade request, i.e. the "new" registrar, can issue a domain object transfer-trade query.

[50] The domain object transfer-trade request allows a new holder of a domain to choose the registrar that he prefers. If he chooses the registrar of the former holder, a domain object trade request issued by that registrar is sufficient. The domain object transfer-trade furthermore allows to pay only once for the domain object transfer-trade instead of paying twice for a domain object transfer followed by domain object trade (or vice-versa).

### 3.2.14 Object transfer-trade cancellation

An object transfer-trade cancellation does only apply to domain objects. A registrar can issue a domain object transfer-trade cancellation in order to cancel a pending domain object transfer-trade that he previously requested by means of a domain object transfer-trade request. A domain object transfer-trade cancellation cannot be issued for a domain object transfer-trade request that has already been executed by the registry.

#### 3.2.14.1 Domain object transfer-trade cancellation

The domain object transfer-trade cancellation procedure is the same as the domain object trade cancellation procedure (cf. section 3.2.7.1). Note that only the registrar that issued the domain object transfer-trade request, i.e. the "new" registrar, can issue a domain object transfer-trade cancellation.

### 3.2.15 Object transfer-restore request

An object transfer-restore request does only apply to domain objects. A registrar can issue a domain object transfer-restore request in order to simultaneously transfer and restore a domain object.

#### 3.2.15.1 Domain object transfer-restore request

The domain object transfer-restore request procedure is the same as the domain object transfer request procedure (cf. section 3.2.8.1). A domain object transfer-restore request can only be issued by a registrar that is not currently the owner of the domain object.

A domain object can be transfer-restored if it does not have a serverTransferProhibited or serverRestoreProhibited status value. Furthermore, a domain object can only be transfer-restored if the domain is in quarantine (pendingDelete) and no other transfer-restore has already been requested for the domain object.

### 3.2.16 Contact object copy on transfer-restore request

When requesting a domain transfer-restore the contacts can be requested to be copied from current registrar to the requesting registrar. For privacy reasons that copied data is NOT available to the registrar until the domain transfer-restore has been successfully executed.

The copy of data happens at the time of domain transfer-restore request and the newly created contact object is tagged as copy. This tag gets removed on successful transfer-restore of the domain it is attached to. As long as the tag exists the contact details cannot be querried and the contact cannot be used for any purpose. If the domain transfer-restore fails the contact can be manually deleted before automatic expiration of unused contacts.

When copy on transfer-restore is used for registrants it is not possible to merge multiple identical contact objects to have all domains of the contact associated to a single contact object unless a trade is done.

### 3.2.17 Object transfer-restore query

An object transfer-restore query does only apply to domain objects. A registrar can issue a domain object transfer-restore query in order to get information about a pending domain object transfer-restore that he previously requested by means of a domain object transfer-restore request.

A domain object transfer-restore query cannot be issued for a domain object transfer-restore request that has already been executed by the registry.

#### 3.2.17.1 Domain object transfer-restore query

The domain object transfer-restore query procedure is the same as the domain object trade query procedure (cf. section 3.2.6.1). Note that only the registrar that issued the domain object transfer-restore request, i.e. the "new" registrar, can issue a domain object transfer-restore query.

### 3.2.18 Object transfer-restore cancellation

An object transfer-restore cancellation does only apply to domain objects. A registrar can issue a domain object transfer-restore cancellation in order to cancel a pending domain object transfer-restore that he previously requested by means of a domain object transfer-restore request. A domain object transfer-restore cancellation cannot be issued for a domain object transfer-restore request that has already been executed by the registry.

#### 3.2.18.1 Domain object transfer-restore cancellation

The domain object transfer-restore cancellation procedure is the same as the domain object trade cancellation procedure (cf. section 3.2.7.1). Note that only the registrar that issued the domain object transfer-restore request, i.e. the "new" registrar, can issue a domain object transfer-restore cancellation.

### 3.2.19 Object availability check

A registrar can perform an object availability check in order to determine whether a certain object can be created in the registry repository, i.e. an object availability check provides a hint that allows a registrar to anticipate the success or failure of creating an object. The check does only consider the object identifier as a criterion. The object identifier is object-specific, i.e. a domain object is identified by its domain name, a host object is identified by its host name and a contact object is identified by its contact ID. In general, an object can be created if there does not already exist an object with the same identifier in the repository (for domain objects) or in the registrar's repository view (for host and contact objects).

#### 3.2.19.1 Domain object availability check

A domain object availability check allows a registrar to determine whether a certain domain object, identified by its domain name, can be created in the registry repository. As described in section 3.2.1.1, a domain object can be created if there does not already exist a domain object with the same domain name in the registry repository, if the domain name is not in the list of domain names that are rejected by the registry due to some special reason and if the domain name respects the well-known domain name rules. In addition a hint is provided telling if registration of a domain requires special attention as for village names.

#### 3.2.19.2 Host object availability check

A host object availability check allows a registrar to determine whether a certain host object, identified by its host name, can be created in the registry repository. As described in section 3.2.1.2, a host object can be created if there does not already exist a host object with the same host name in the registrar's repository view.

#### 3.2.19.3 Contact object availability check

A contact object availability check allows a registrar to determine whether a certain contact object, identified by its contact ID, can be created in the registry repository. As described in section 3.2.1.3, a contact object can be created if there does not already exist a contact object with the same contact ID in the registrar's repository view.

### 3.2.20 Object information retrieval

A registrar can perform an object information retrieval in order to get information about a certain object, identified by its object identifier, that exists in his repository view [51].

#### 3.2.20.1 Domain object information retrieval

A domain object information retrieval allows a registrar to get information about a certain domain object that is identified by its domain name. Information can also be retrieved for domain objects that have been requested to be created but that do not yet exist (i.e. for domain objects that are in state pendingCreate) and for domain objects that are in quarantine (i.e. for domain objects that are in state pendingDelete). The type of information provided to the registrar will be described in later chapters, but we already mentioned that a registrar cannot see whether a domain object transfer (or transfer-trade or transfer-restore) has been requested for a domain object in his repository view. If a registrar that does not own a domain requests information about a domain only limited information will be provided, especially telling wether the domain is pendingDelete or the requesting registrar has a pending transfer, transfer-trade or transfer-restore operation on the domain.

#### 3.2.20.2 Host object information retrieval

A host object information retrieval allows a registrar to get information about a certain host object that is identified by its host name. The type of information provided to the registrar will be described in later chapters.

#### 3.2.20.3 Contact object information retrieval

A contact object information retrieval allows a registrar to get information about a certain contact object that is identified by its contact ID. The type of information provided to the registrar will be described in later chapters.

### 3.2.21 Automatic contact copy

When requesting a domain transfer or domain transfer-restore original contact objects (either contact or holder) can be copied from the sponsoring registrar's account. To satisfy data privacy, the copied contact is not accessible by the requesting registrar until the matching transfer or transfer-restore has been successfully executed. (The contact ID is reserved though any contact:info or attempt to link the contact to another domain will return an error). The registrar interface also hides all sensitive data of those contacts.

Automatic copy can be requested by prepending the contact ID with the 'a' (Unicode 00AA, "FEMININE ORDINAL INDICATOR"). Prior to transfer or transfer-restore request the contact ID may NOT be in use. When processing the request the contact is automatically created and

[51] Please note that a registrar cannot retrieve information about objects that exist in another registrar's repository view.

initialized with the details of the contact in the sponsoring registrar's view. Thus, any change done to the contact by the sponsoring registrar between transfer/transfer-restore request and approval will be not be retained by the auto-copy feature.

When the transfer/transfer-restore operation is aborted (timed out, rejected or canceled) the contact object remains in the requesting registrar's view (and cannot be used or querried with contact:info) until it gets automatically removed because of long-lasting unused state or until the registrar deletes the contact.

Until approval and execution of domain transfer/transfer-restore the contact object is locked and cannot be linked to any other domain name. Once the transfer/transfer-restore operation has been executed the contact can be used as any other contact.

## 3.3 Summary of the EPP commands: pending as opposed to non-pending actions

The table below states for each EPP command whether the action corresponding to the command is executed immediately by the EPP server or whether the action is pending. For those commands that are executed immediately, the EPP client, i.e. the registrar, will immediately get a response from the EPP server that states whether the action was executed successfully or not. For those commands that are pending, the EPP client will also immediately get a response from the EPP server but this response only states that the command has been registered. A response that states whether the action has been successfully executed or not will be provided to the EPP client by means of a poll-message at a later moment.

| **EPP command** | **Processing** |
|---|---|
| EPP hello | Executed immediately (greeting in response). |
| EPP startTLS | Executed right after successful response. |
| EPP login | Executed immediately. |
| EPP logout | Executed immediately. |
| EPP poll request | Executed immediately. |
| EPP poll ack | Executed immediately. |
| EPP domain create | Pending. |
| EPP domain update | Pending **if**:<br>The update switches the domain from reserved to active<br>**Or** hosts are added/removed to/from the domain and domain is/remains active after the update. |

| | |
|---|---|
| EPP domain delete immediate | Executed immediately. Even if the domain object will not be deleted immediately, but only after the quarantine period, the domain object is immediately set to pendingDelete (not exported to the ".lu" name servers anymore), and therefore one can consider this action as executed immediately. |
| EPP domain delete setDate | Pending. The domain object will be set to pendingDelete at the specified date. The registrar will be notified by a poll-message at that moment. |
| EPP domain delete cancel | Executed immediately. |
| EPP domain restore | Executed immediately. |
| EPP domain trade request | Pending. |
| EPP domain trade query | Executed immediately. |
| EPP domain trade cancel | Executed immediately. |
| EPP domain transfer request | Pending. |
| EPP domain transfer query | Executed immediately. |
| EPP domain transfer cancel | Executed immediately. |
| EPP domain transfer-trade request | Pending. |
| EPP domain transfer-trade query | Executed immediately. |
| EPP domain transfer-trade cancel | Executed immediately. |
| EPP domain transfer-restore request | Pending. |
| EPP domain transfer-restore query | Executed immediately. |
| EPP domain transfer-restore cancel | Executed immediately. |
| EPP domain check | Executed immediately. |
| EPP domain info | Executed immediately. |
| EPP host create | Executed immediately. |
| EPP host update | Pending **if** the command contains an addition, removal or change of an IP address and a glue record is necessary. |
| EPP host delete | Executed immediately. |
| EPP host check | Executed immediately. |
| EPP host info | Executed immediately. |
| EPP contact create | Executed immediately. |

| | |
|---|---|
| EPP contact update | Executed immediately. |
| EPP contact delete | Executed immediately. |
| EPP contact check | Executed immediately. |
| EPP contact info | Executed immediately. |

## 3.4 Additional policy issues

### 3.4.1 Registrar credit and chargeable actions

For the transactions that are initiated by an EPP command issued by a registrar, the registry system checks whether the registrar's credit allows the transaction when it gets the command. If the registrar's credit does not allow the transaction, the command fails. For those transactions that require a pending processing, the registrar's credit is decremented when the action is executed, i.e. after the pending processing. At that moment, the registrar's credit is decremented even if the registrar's credit goes below 0. This also applies to the renewal of a domain.

The fees for the different transactions are defined in the contractual framework.

The credit and transaction fees are always displayed excluding VAT in the registry system. If a registrar has to pay VAT it will be deducted from the credit before introduction into the registry system. VAT details will be mentionned on the monthly registrar invoice with summary of the transactions executed during the previous month.

# Part III

# Registry system specification

# Chapter 4

# Registry system overview

## 4.1 Overview

The DNS-LU registry is composed of multiple software entities. Visible to the registrars there are:

**EPP Server**

This entity supports EPP protocol (see chapter 5 about EPP Server and EPP protocol) and allows all domain, contact and host related operations.

**Registrar Interface**

This entity is a web-based interface (see chapter 6) that allows registrars to check their credit state, list their domains, contact and hosts.

Execution of EPP operations via basic web-forms is planned for the Registrar Interface but not available as of now. Registrars should not rely on this planned feature.

**Whois**

The WHOIS service is not restricted to registrars only and provides a snapshot of the registry information. This snapshot is updated twice a day at the time of writing.

**NOTE**: WHOIS should only be used to determine current holder of existing domains when needed, not in order to determine wether a domain is available. Use EPP domain check command for this. (EPP domain info provides only a subset of the information available in WHOIS for domains managed by other registrars).

**DNS Server(s)**

The DNS servers are not restricted to registrars only and provides a snapshot of the registry information. This snapshot is updated four times a day, every 6 hours, at the time of writing.

# Chapter 5

# EPP server specification

The EPP Server offers the interface for registrars to manage their domain's data. It uses the EPP protocol as defined in rfc3730[2], rfc3731[3], rfc3732[4] and rfc3733[5]. In addition to those RFC there are some DNS-LU specific extensions and restrictions.

The EPP server offers two TCP-based transports for communication.

The first one, DNS-LU specific is a raw **UTF-8** encoded stream over TCP where XML messages are sent an received, the closing tag </epp> being used as delimiter.

The second one is the rfc3734[6] implementation. Here care should be taken regarding the header. Multiple issues were introduced in the implementation regarding the value contained in the header. The RFC states that the header MUST be the size of a complete message (XML and header), implementation did omit the header size. This has been fixed at the time of writing.

For both transport protocols SSL is supported, either in tunneled SSL mode starting or on demand for rfc3734[6] transport using the DNS-LU extension startTLS.

When receiving an EPP command it is first validated against the EPP schemas and only processed further if validation is successful. If there are validation errors, the response will have result code 2001 (syntax error) or 2400 (command failed). On successful validation each command is checked for consistency and errors are reported as appropriate using standard EPP result codes. Each result element in the response contains some description if available which should provide some help in tacking the error.

**Note:** The EPP Server does not require a specific formatting of EPP XML messages nor does it assume the clients have any such restriction. As long as the messages are welformed XML and satisfy the schema client and server are expected to be able to parse the messages successfully.

With exception of a few elements in contact related messages or for IDN domains all values must be printable US-ASCII.

## 5.1 DNS-LU Schema extensions

To fit DNS-LU policy needs, extensions were added to EPP for various objects using the extension model defined for EPP. This schema affects contact operations, poll messages and most of the domain operations. These will be explained in the following sections of this chapter in more depth. (See also chapter 3 on object policies)

**Note:** In the samples in following chapters most possible fields are listed, read the EPP RFCs and field listings below commands to see restrictions on concurrent use. Looking at the different XML schemas is also recommended.

## 5.2 Beware: Policy Limitations

This documentation is a technical documentation trying to be as comprehensive as possible. However, policy limitations maybe in place that limit some of the functions. The two most notable points are:

- **IDNs:** although described in this document, international domain names are not allowed until a new policy is in place. The technical implementation is not likely to change, though.
- **Status Flags:** client* status flags are described, but currently not available, except for clientHold and clientRenewProhibited which may be set.
- **SSL and TLS:** although described in this document and supported by the software it may not be available yet. See news on DNS-LU homepage to check when it is made available.

## 5.3 Object-independent EPP commands

### 5.3.1 EPP hello command

This command follows standard EPP, rfc3730[2]. No DNS-LU extensions or restrictions.

#### Client request

At connection initiation the server processes an implicit `hello` command and sends a greeting to the client. This greeting allows the server to inform the client about the supported protocol parts and extensions, languages and which server the client is talking to. In addition to this implicit hello command at TCP connection establishment the EPP client may send additional `hello` commands at any time.

The EPP server responds with a greeting message that before login displays the server capabilities (the capabilities represent the EPP protocol modules that are supported, what extensions are supported, what versions of each are supported). After a successful login the claimed capacities are reduced to the ones the client requested at login. Same principle applies for the message language.

```
<?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"
   <hello/>
</epp>
```

Sample hello request

#### Server response

The EPP Server responds with a greeting toa hello command.

If the client is already authenticated the server will only list the extensions enabled for the current session, otherwise it will list all the supported extensions. The elements affected by this rule are: `objURI`, `svcExtension+extURI` and `lang`.

**CAUTION**: Disclosure policy information published by EPP Server is not authoritative!

```
<?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>DNS-LU EPP Server epp.dns.lu</svID>
```

```
      <svDate>2005-09-30T13:05:22Z</svDate>
      <svcMenu>
         <version>1.0</version>
         <lang>en</lang>
         <objURI>urn:ietf:params:xml:ns:domain-1.0</objURI>
         <objURI>urn:ietf:params:xml:ns:host-1.0</objURI>
         <objURI>urn:ietf:params:xml:ns:contact-1.0</objURI>
         <svcExtension>
            <extURI>http://www.dns.lu/xml/epp/dnslu-1.0</extURI>
         </svcExtension>
      </svcMenu>
      <dcp>
         <access><none/></access>
         <statement>
            <purpose><prov/></purpose>
            <recipient><ours/></recipient>
            <retention><indefinite/></retention>
         </statement>
      </dcp>
   </greeting>
</epp>
```

Sample EPP greeting response

### 5.3.2 EPP startTLS command

This command is a DNS-LU extension. It's aim is to allow sharing a single TCP port for encrypted and non-encrypted communication.

#### Client request

The client issues a startTLS request to request initiation of TLS session.

```
<?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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:startTLS host="epp-test.dns.lu" />
            <dnslu:clTRID>STLS1</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample startTLS request

All fields must be printable US-ASCII.

**host** The hostname to which connection happend
This is the host name that is expected in certificate commonName attribute.

#### Server response

The server answers with result code 1000 if TLS is supported. Right after this response the TLS handshake is started. If the handshake fails the connection is closed without further notice.

On any other result code the session continues normally without starting SSL session. If the session is already in SSL mode the server answers with result code 2002, if the selected transport protocol does not support TLS result code 2101 is returned.

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>4516E89-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample poll response with 4 pending messages

### 5.3.3 EPP login command

This command follows standard EPP, rfc3730[2]. No DNS-LU extensions or restrictions.

#### Client request

The client authenticates itself to the EPP server for the rest of the TCP session with a `login` command. This command also allows updating the client's authentication password. In the login request the client chooses the message language, EPP protocol modules and extensions (from the list provided by server during the greeting) which it wants to use during the session.

```
<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>
      <login>
         <clID>username</clID>
         <pw>password</pw>
         <newPW>newPassword</newPW>
         <options>
            <version>1.0</version>
            <lang>en</lang>
         </options>
         <svcs>
            <objURI>urn:ietf:params:xml:ns:domain-1.0</objURI>
            <objURI>urn:ietf:params:xml:ns:host-1.0</objURI>
            <objURI>urn:ietf:params:xml:ns:contact-1.0</objURI>
            <svcExtension>
               <extURI>http://www.dns.lu/xml/epp/dnslu-1.0</extURI>
            </svcExtension>
         </svcs>
      </login>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample login request

All fields must be printable US-ASCII, lang, objURI and extURI must be values advertised in greeting.

| | |
|---|---|
| **clID** | Client username |
| **pw** | Client (current) password |
| **newPW** | (optional) new password |
| **version** | Requested EPP protocol version (must be '1.0') |
| **lang** | Requested language for textual messages in communication (must be 'en') |

| | |
|---|---|
| **objURI** | Standard EPP protocol module (domain, contact and host recommended) |
| **extURI** | Extension modules to EPP protocol (dns-lu should be provided) |
| **clTRID** | Client transaction ID |

**Server response**

On a successful login (with optional password update) request the EPP server answers with a result code of 1000 and opens access to other operations for the client.

On a failed login request, or failed password update, the EPP server responds with a result code of 2200 and waits for a new login request. The server may close the connection after too many failed logins.

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>680C3E56-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample success response

### 5.3.4 EPP logout command

This command follows standard EPP, rfc3730[2]. No DNS-LU extensions or restrictions.

**Client request**

The client issues a `logout` request to terminate it's session gracefully.

```
<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>
      <logout/>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample logout request

All fields must be printable US-ASCII.

| | |
|---|---|
| **clTRID** | Client transaction ID |

**Server response**

The server answers with result code 1500 and closes the connection after a `logout` request.

```
<?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">
   <response>
      <result code="1500">
```

```
         <msg>Command completed successfully; ending session</msg>
      </result>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>680C3E57-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample success response

### 5.3.5 EPP poll request command

This command follows standard EPP, rfc3730[2]. No DNS-LU extensions or restrictions for the request. The response contains DNS-LU formatted poll message data.

#### Client request

The client issues a poll request of type 'req' to query the poll message queue.

```
<?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 op="req"/>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample poll-req request

All fields must be printable US-ASCII.

**clTRID** Client transaction ID

#### Server response

The server answers with result code 1300 if the poll message queue is empty.

If the queue has at least one message the server answers with a result code of 1301 and then the oldest message in the queue is put into the response with it's ID and the total count of poll messages in the queue. The content of the oldest poll message is formated in a DNS-LU specific layout.

```
<?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">
   <response>
      <result code="1301">
         <msg>Command completed successfully; ack to dequeue</msg>
      </result>
      <msgQ count="4" id="1">
         <qDate>2005-10-03T07:55:13Z</qDate>
         <msg lang="en">
            <dnslu:pollmsg type="1234" xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                   xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
               <dnslu:roid>D123-DNSLU</dnslu:roid>
               <dnslu:object>mydomain.lu</dnslu:object>
               <dnslu:clTRID>89ABCDEF</dnslu:clTRID>
               <dnslu:svTRID>13868389</dnslu:svTRID>
               <dnslu:exDate>2005-10-05T07:37:10Z</dnslu:exDate>
               <dnslu:ns name="ns.domain.lu">Test failed</dnslu:ns>
               <dnslu:reason>Because!</dnslu:reason>
```

```
            <dnslu:extra name="field">some extra information<dnslu:extra>
         </dnslu:pollmsg>
      </msg>
   </msgQ>
   <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>4516E89-DNSLU</svTRID>
   </trID>
</response>
</epp>
```

Sample poll response with 4 pending messages

All fields are printable US-ASCII.

| | |
|---|---|
| **pollmsg@type** | Poll message type ID |
| **pollmsg/roid** | Registry Object ID of affected object |
| **pollmsg/object** | Client ID for affected object (domain name, host name, contact ID) |
| **pollmsg/clTRID** | (optional) Client transaction ID of inducing request |
| **pollmsg/svTRID** | (optional) Server transaction ID of inducing request |
| **pollmsg/exDate** | (optional) Expiration date of created domain |
| **pollmsg/ns** | (optional, unbounded) Nameserver related message |
| **ns@name** | Name of affected host (all for all hosts) |
| **pollmsg/reason** | (optional) Reason of domain rejection |
| **pollmsg/extra** | (optional, unbounded) Extra information |
| **extra@name** | Key to the value |
| **clTRID** | Client transaction ID |
| **svTRID** | Server transaction ID |

### Poll message types

To make processing of poll messages easy for automated systems a set of poll messages type IDs has be defined[1]:

| ID | Description |
|---|---|
| 0 | Unknown type |
| 1 | Domain Transfer executed |
| 2 | Domain Transfer rejected by holder |
| 3 | Domain Transfer aborted by registry |
| 4 | Domain TransferTrade executed |
| 5 | Domain TransferTrade rejected by holder |
| 6 | Domain TransferTrade aborted by registry |
| 7 | Domain Trade executed |
| 8 | Domain Trade rejected by holder |
| 9 | Domain Trade aborted by registry |

[1] This list is not frozen and may be extended in future as required

| | |
|---|---|
| 10 | Domain TransferRestore executed |
| 11 | Domain TransferRestore rejected by holder |
| 12 | Domain TransferRestore aborted by registry |
| 13 | Active domain creation executed |
| 14 | Reserved domain creation executed |
| 15 | Domain creation rejected |
| 16 | Domain deleted (sent to quarantine) |
| 17 | Unlinked host deleted |
| 18 | Unlinked holder deleted |
| 19 | Unlinked contact deleted |
| 20 | Pending domain update executed |
| 21 | Pending domain update failed |
| 22 | Domain renewed |
| 23 | Domain deleted from quarantine |
| 24 | Pending host update executed |
| 25 | Pending host update failed |
| 26 | Domain transferred away |
| $> 26$ | Reserved for future use |

### 5.3.6 EPP poll acknowledgement command

This command follows standard EPP, rfc3730[2]. No DNS-LU extensions or restrictions.

#### Client request

The client issues a poll request of type 'ack' to acknowledge the oldest poll message in the queue, making the following poll message available for retrieval. The response indicates the count of remaining poll messages.

#### A) Command syntax

```
<?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 op="ack" msgID="1"/>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample poll-ack request

All fields must be printable US-ASCII.

| | |
|---|---|
| **@msgID** | ID of poll message to acknoledge |
| **clTRID** | Client transaction ID |

**Server response**

The server answers with result code 1300 if the poll message queue is empty and acknowledging was successful.

If the queue has at least one message left after aknowledging the server answers with a result code of 1000 indicating the count of poll messages remaining in the queue as well as the ID of the acknowledged poll message.

```
<?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">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <msgQ count="3" id="1"/>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>4516E8A-DNSLU</svTRID>
    </trID>
  </response>
</epp>
```

Sample poll-ack response with 3 messages left

## 5.4 Contact-specific EPP commands and corresponding procedures

This section describes the EPP commands that apply to contact objects, as specified in rfc3733[5].

DNS-LU internally makes a difference between holder contacts and administrative or technical contacts. This shows up in the EPP protocol by an additionnal type element. In addition DNS-LU handles disclosure settings differently from what is supported by EPP. Each disclosure flag can be set to 1 to force showing the info or 0 to force hiding the info. If a flag is not explicitely mentioned the DNS-LU default at export time will be used. The disclosure flags may be added or removed with the update command, allowing registrars to reset settings to DNS-LU default.

The DNS-LU holders are contacts with some details stripped. Stripped details include phone number and fax number. Email must be provided for holder as of EPP schema but are ignored by DNS-LU. A holder is either moral person (organization) or physical person and must not have an organization field.

Postal info is always managed in local mode, accepting latin1 part of Unicode. Optional field `contact:sp` is not accepted by DNS-LU, optional field `contact:pc` is required by DNS-LU. At least 1 street field must be provided, at most 3.

### 5.4.1 EPP check command

This command follows standard EPP - Contact mapping, rfc3733[5]. No DNS-LU extensions or restrictions.

#### Client request

The client issues a contact check request to determine if contact IDs are available for new creations or if they are already in use.

```
<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>
         <contact:check xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
               xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0␣contact-1.0.xsd">
            <contact:id>H100</contact:id>
            <contact:id>C100<contact:id>
            <contact:id>C101<contact:id>
         </contact:check>
      </check>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample contact check request

All fields must be printable US-ASCII.

| | |
|---|---|
| **id** | (unbounded) Contact ID to check |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 1000 on successfull processing of the request.
For each requested contact ID the server indicates if the ID is available for creating new contacts.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:iana:xml:ns:epp"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="urn:iana:xml:ns:epp␣epp.xsd">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <resData>
         <contact:chkData xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
               xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0␣contact-si-1.0.xsd">
            <contact:cd>
               <contact:id avail="0">H100</contact:id>
            </contact:cd>
            <contact:cd>
               <contact:id avail="0">C100</contact:id>
            </contact:cd>
            <contact:cd>
               <contact:id avail="1">C101</contact:id>
            </contact:cd>
         </contact:chkData>
      </resData>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>47A24FA2-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample contact check response

### 5.4.2 EPP info command

This command follows standard EPP - Host mapping, rfc3733[5]. DNS-LU extension adds modified disclosure setting handling, differentiation between contacts and holders.

#### Client request

The client issues a contact info request to fetch a contact's or holder's details.

```
<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>
         <contact:info xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0 contact-1.0.xsd">
            <contact:id>C100</contact:id>
         </contact:info>
      </info>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample contact/holder info request

All fields must be printable US-ASCII.

| | |
|---|---|
| **id** | ID of contact/holder whose info to fetch |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 2303 if a contact/holder with the given ID did not exist.

On successful processing of the contact/holder info request a result code of 1000 is returned and a contact:infData item provides detailled information about the contact/holder object.

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <resData>
         <contact:infData xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0 contact-1.0.xsd">
            <contact:id>H0008</contact:id>
            <contact:roid>H3-DNSLU</contact:roid>
            <contact:status s="ok"/>
            <contact:status s="linked"/>
            <contact:postalInfo type="loc">
               <contact:name>Fondation RESTENA</contact:name>
               <contact:addr>
                  <contact:street>6, rue Coudenhove-Kalergi</contact:street>
                  <contact:city>Luxembourg</contact:city>
                  <contact:pc>1359</contact:pc>
                  <contact:cc>LU</contact:cc>
               </contact:addr>
            </contact:postalInfo>
            <contact:email>dummy@dns.lu</contact:email>
            <contact:clID>restena-id</contact:clID>
            <contact:crID>restena-id</contact:crID>
            <contact:crDate>2005-10-05T07:37:10Z</contact:crDate>
            <contact:upID>restena-id</contact:crID>
            <contact:upDate>2005-11-17T12:59:11Z</contact:crDate>
         </contact:infData>
      </resData>
```

```
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/200/10/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:infData>
                  <dnslu:contact>
                     <dnslu:type>holder_org</dnslu:type>
                     <dnslu:disclose>
                        <dnslu:name flag="1"/>
                        <dnslu:addr flag="0"/>
                     </dnslu:disclose>
                  </dnslu:contact>
               </dnslu:infData>
            </dnslu:resData>
         </dnslu:ext>
      </extension>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>525907E0-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample holder info response

Allowed characters for all field are printable Latin1, except contact:id, contact:email and clTRID which are US-ASCII.

| | |
|---|---|
| **id** | Holder ID |
| **roid** | Registry unique ID for the holder |
| **status** | (unbounded) status flags as defined by EPP |
| **name** | Holder name |
| **street** | Holder address, street (1 to 3 times) |
| **city** | Holder address, city |
| **pc** | Holder address, zip code |
| **cc** | Holder country |
| **email** | ignored |
| **clID** | Registrar who owns the holder object |
| **crID** | registrar who created the holder |
| **crDate** | creation date of the holder |
| **upID** | (optional) Registrar who did last update to the holder |
| **upDate** | (optional) Date of last update to the holder |
| **type** | (extension) Must be holder_pers oder holder_org |
| **disclose/name** | (extension, optional) Show/hide name in WHOIS |
| **disclose/addr** | (extension, optional) Show/hide address in WHOIS |
| **clTRID** | Client transaction ID |

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <resData>
         <contact:infData xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0␣contact-1.0.xsd">
            <contact:id>C0001</contact:id>
            <contact:roid>C3-DNSLU</contact:roid>
            <contact:status s="ok"/>
            <contact:status s="linked"/>
```

```
            <contact:postalInfo type="loc">
            <contact:name>Bruno Prémont</contact:name>
            <contact:org>Fondation RESTENA</contact:org>
               <contact:addr>
                  <contact:street>6, rue Coudenhove-Kalergi</contact:street>
                  <contact:city>Luxembourg</contact:city>
                  <contact:pc>1359</contact:pc>
                  <contact:cc>LU</contact:cc>
               </contact:addr>
            </contact:postalInfo>
            <contact:voice>+352.42440926</contact:voice>
            <contact:email>bruno.premont@restena.lu</contact:email>
            <contact:clID>restena-id</contact:clID>
            <contact:crID>restena-id</contact:crID>
            <contact:crDate>2005-10-05T09:24:38Z</contact:crDate>
            <contact:upID>restena-id</contact:crID>
            <contact:upDate>2005-11-17T10:31:47Z</contact:crDate>
         </contact:infData>
      </resData>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/200/10/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:infData>
                  <dnslu:contact>
                     <dnslu:type>contact</dnslu:type>
                     <dnslu:disclose>
                        <dnslu:name flag="0"/>
                        <dnslu:addr flag="0"/>
                        <dnslu:voice flag="1"/>
                        <dnslu:fax flag="1"/>
                        <dnslu:email flag="0"/>
                     </dnslu:disclose>
                  </dnslu:contact>
               </dnslu:infData>
            </dnslu:resData>
         </dnslu:ext>
      </extension>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>525907E2-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample contact info response

Allowed characters for all field are printable Latin1, except contact:id, contact:email and clTRID which are US-ASCII.

| | |
|---|---|
| **id** | Contact ID |
| **roid** | Registry unique ID for the contact |
| **status** | (unbounded) status flags as defined by EPP |
| **name** | Contact name |
| **org** | (optional) Contact's organisation |
| **street** | Contact address, street (1 to 3 times) |
| **city** | Contact address, city |
| **pc** | Contact address, zip code |
| **cc** | Contact country |
| **email** | Contact email |
| **voice** | (optional) Contact voice |
| **fax** | (optional) Contact fax |
| **clID** | Registrar who owns the contact object |
| **crID** | registrar who created the contact |
| **crDate** | Creation date of the contact |

| | |
|---|---|
| **upID** | (optional) Registrar who did last update to the contact |
| **upDate** | (optional) Date of last update to the contact |
| **type** | (extension) 'contact' |
| **disclose/name** | (extension, optional) Show/hide name in WHOIS |
| **disclose/addr** | (extension, optional) Show/hide address in WHOIS |
| **disclose/voice** | (extension, optional) Show/hide phone number in WHOIS |
| **disclose/fax** | (extension, optional) Show/hide fax number in WHOIS |
| **disclose/email** | (extension, optional) Show/hide email address in WHOIS |
| **clTRID** | Client transaction ID |

### 5.4.3 EPP create command

This command follows standard EPP - Contact mapping, rfc3733[5]. DNS-LU extension adds modified disclosure setting handling and differentiation between contacts and holders.

DNS-LU limitations on rfc3731[3] do exist as well. PostalInfo must be provided exactly once with type local. Unicode subset allowed for fields is latin1. For holder contacts phone, fax and organisation elements may not be provided and email element is ignored. Authinfo is ignored for contacts and holders.

#### Client request

The client issues a holder create request to create a holder.

```
<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>
         <contact:create xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0␣contact-1.0.xsd">
            <contact:id>H100</contact:id>
            <contact:postalInfo type="loc">
               <contact:name>Fondation RESTENA</contact:name>
               <contact:addr>
                  <contact:street>6, rue Coudenhove-Kalergi</contact:street>
                  <contact:city>Luxembourg</contact:city>
                  <contact:pc>1359</contact:pc>
                  <contact:cc>LU</contact:cc>
               </contact:addr>
            </contact:postalInfo>
            <contact:email>dummy@dnslu.lu</contact:email>
            <contact:authInfo>
               <contact:pw>dummy</contact:pw>
            </contact:authInfo>
         </contact:create>
      </create>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:create>
               <dnslu:contact>
                  <dnslu:type>holder_org</dnslu:type>
                  <dnslu:disclose>
                     <dnslu:name flag="1"/>
                     <dnslu:addr flag="0"/>
                  </dnslu:disclose>
               </dnslu:contact>
            </dnslu:create>
         </dnslu:ext>
      </extension>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample holder create request

All fields must be printable Latin1, except contact:id and clTRID which must be US-ASCII.

| | |
|---|---|
| **id** | Holder ID |
| **name** | Holder name |
| **street** | Holder address, street (1 to 3 times) |
| **city** | Holder address, city |
| **pc** | Holder address, zip code |
| **cc** | Holder country |
| **email** | ignored |
| **authInfo/pw** | ignored |
| **type** | (extension) Must be holder_pers oder holder_org |
| **disclose/name** | (extension, optional) Show/hide name in WHOIS |
| **disclose/addr** | (extension, optional) Show/hide address in WHOIS |
| **clTRID** | Client transaction ID |

The client issues a contact create request to create a contact.

```
<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>
         <contact:create xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0␣contact-1.0.xsd">
            <contact:id>C100</contact:id>
            <contact:postalInfo type="loc">
               <contact:name>Bruno Prémont</contact:name>
               <contact:org>Fondation RESTENA</contact:org>
               <contact:addr>
                  <contact:street>6, rue Coudenhove-Kalergi</contact:street>
                  <contact:city>Luxembourg</contact:city>
                  <contact:pc>1359</contact:pc>
                  <contact:cc>LU</contact:cc>
               </contact:addr>
            </contact:postalInfo>
            <contact:voice>+352.42440928</contact:voice>
            <contact:fax>+352.42440928</contact:fax>
            <contact:email>bruno.premont@restena.lu</contact:email>
         </contact:create>
      </create>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:create>
               <dnslu:contact>
                  <dnslu:type>contact</dnslu:type>
                  <dnslu:disclose>
                     <dnslu:name flag="1"/>
                     <dnslu:addr flag="0"/>
                     <dnslu:voice flag="0"/>
                     <dnslu:email flag="0"/>
                  </dnslu:disclose>
               </dnslu:contact>
            </dnslu:create>
         </dnslu:ext>
      </extension>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample contact create request

All fields must be printable Latin1, except contact:id, contact:email and clTRID which must be US-ASCII.

| | |
|---|---|
| **id** | Contact ID |
| **name** | Contact name |
| **org** | (optional) Contact's organisation |
| **street** | Contact address, street (1 to 3 times) |
| **city** | Contact address, city |
| **pc** | Contact address, zip code |
| **cc** | Contact country |
| **email** | Contact email |
| **voice** | (optional) Contact voice |
| **fax** | (optional) Contact fax |
| **authInfo/pw** | ignored |
| **type** | (extension) Must be contact |
| **disclose/name** | (extension, optional) Show/hide name in WHOIS |
| **disclose/addr** | (extension, optional) Show/hide address in WHOIS |
| **disclose/voice** | (extension, optional) Show/hide phone number in WHOIS |
| **disclose/fax** | (extension, optional) Show/hide fax number in WHOIS |
| **disclose/email** | (extension, optional) Show/hide email address in WHOIS |
| **clTRID** | Client transaction ID |

### Server response

The server answers with result code 2302 if a contact/holder with the given ID already existed.

On successful creation of the new contact/holder a result code of 1000 is returned and a contact:creData item provides limited information about the creation.

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <resData>
         <contact:creData xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0␣contact-1.0.xsd">
            <contact:id>H0008</contact:id>
            <contact:crDate>2005-10-05T07:37:10Z</contact:crDate>
         </contact:creData>
      </resData>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>1C438265-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample holder/contact create response

All fields are printable US-ASCII.

| | |
|---|---|
| **id** | Contact/Holder ID |
| **crDate** | UTC creation data of the object |
| **clTRID** | Client transaction ID |
| **svTRID** | Server transaction ID |

### 5.4.4 EPP update command

This command follows standard EPP - Contact mapping, rfc3733[5]. DNS-LU extension adds modified disclosure setting handling and differentiation between contacts and holders.

DNS-LU limitations on rfc3731[3] do exist as well. PostalInfo must be provided exactly once with type local. The allowed Unicode subset for fields is printable latin1. For holder contacts phone, fax and organisation elements may not be provided and email element is ignored. Authinfo is not allowed for contacts and holders. Status updates (clientUpdateProhibited, clientDeleteProhibited) are not avilable at this time.

#### Client request

The client issues a holder upate request to modify a holder's details.

```
<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>
         <contact:update xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0␣contact-1.0.xsd">
            <contact:id>H100</contact:id>
            <contact:chg>
               <contact:postalInfo type="loc">
                  <contact:name>Gilles Massen</contact:name>
                  <contact:addr>
                     <contact:street>Building A</contact:street>
                     <contact:street>Department X</contact:street>
                     <contact:street>rue de Luxembourg 10</contact:street>
                     <contact:city>Luxembourg</contact:city>
                     <contact:pc>1359</contact:pc>
                     <contact:cc>LU</contact:cc>
                  </contact:addr>
               </contact:postalInfo>
            </contact:chg>
         </contact:update>
      </update>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:update>
               <dnslu:contact>
                  <dnslu:add>
                     <dnslu:disclose>
                        <dnslu:name flag="0"/>
                        <dnslu:addr flag="1"/>
                     </dnslu:disclose>
                  </dnslu:add>
                  <dnslu:rem>
                     <dnslu:disclose>
                        <dnslu:name flag="1"/>
                        <dnslu:addr flag="1"/>
                     </dnslu:disclose>
                  </dnslu:rem>
               </dnslu:contact>
            </dnslu:update>
         </dnslu:ext>
      </extension>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample holder update request

All fields must be printable Latin1, except contact:id and clTRID which must be US-ASCII.

**id** Holder ID

| | |
|---|---|
| **name** | (optional) Holder name |
| **addr** | (optional) |
| **street** | Holder address, street (1 to 3 times) |
| **city** | Holder address, city |
| **pc** | Holder address, zip code |
| **cc** | Holder country |
| **email** | (not allowed) |
| **authInfo/pw** | (not allowed) |
| **disclose/name** | (extension, optional) Show/hide name in WHOIS |
| **disclose/addr** | (extension, optional) Show/hide address in WHOIS |
| **clTRID** | Client transaction ID |

If provided, the addr element must have complete contents as for holder creation. A single disclosure flag may not be added and removed at same time (Note: the sample shows all disclosure flags below 'add' and 'rem' though when used each disclosure may apear at most in one of both).

CAUTION: Modifying holder flags is not allowed.

The client issues a holder upate request to modify a contact's details.

```
<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>
         <contact:update xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0␣contact-1.0.xsd">
            <contact:id>C100</contact:id>
            <contact:chg>
               <contact:postalInfo type="loc">
                  <contact:name>Gilles Massen</contact:name>
                  <contact:org>Fondation RESTENA</contact:org>
                  <contact:addr>
                     <contact:street>Building A</contact:street>
                     <contact:street>Department X</contact:street>
                     <contact:street>rue de Luxembourg 10</contact:street>
                     <contact:city>Luxembourg</contact:city>
                     <contact:pc>1359</contact:pc>
                     <contact:cc>LU</contact:cc>
                  </contact:addr>
               </contact:postalInfo>
               <contact:voice>+352.42440925</contact:voice>
               <contact:fax>+352.42440925</contact:fax>
               <contact:email>massen@restena.lu</contact:email>
            </contact:chg>
         </contact:update>
      </update>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:update>
               <dnslu:contact>
                  <dnslu:add>
                     <dnslu:disclose>
                        <dnslu:name flag="1"/>
                        <dnslu:org flag="0"/>
                        <dnslu:addr flag="1"/>
                        <dnslu:voice flag="0"/>
                        <dnslu:fax flag="1"/>
                        <dnslu:email flag="0"/>
                     </dnslu:disclose>
                  </dnslu:add>
                  <dnslu:rem>
                     <dnslu:disclose>
                        <dnslu:name flag="1"/>
                        <dnslu:org flag="0"/>
                        <dnslu:addr flag="1"/>
                        <dnslu:voice flag="0"/>
```

```
                        <dnslu:fax flag="1"/>
                        <dnslu:email flag="0"/>
                     </dnslu:disclose>
                  </dnslu:rem>
               </dnslu:contact>
            </dnslu:update>
         </dnslu:ext>
      </extension>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample contact update request

All fields must be printable Latin1, except contact:id and clTRID which must be US-ASCII.

| | |
|---|---|
| **id** | Holder ID |
| **name** | (optional) Contact name |
| **org** | (optional) Campany/Organization name |
| **addr** | (optional) Address block |
| **street** | Contact address, street (1 to 3 times) |
| **city** | Contact address, city |
| **pc** | Contact address, zip code |
| **cc** | Contact country |
| **voice** | (optional) Contact phone number |
| **fax** | (optional) Contact fax number |
| **email** | (optional) Contact email address |
| **authInfo/pw** | (not allowed) |
| **disclose/name** | (extension, optional) Show/hide name in WHOIS |
| **disclose/addr** | (extension, optional) Show/hide address in WHOIS |
| **disclose/voice** | (extension, optional) Show/hide phone number in WHOIS |
| **disclose/fax** | (extension, optional) Show/hide fax number in WHOIS |
| **disclose/email** | (extension, optional) Show/hide email address in WHOIS |
| **clTRID** | Client transaction ID |

If provided, the addr element must have complete contents as for holder creation. A single disclosure flag may not be added and removed at same time (Note: the sample shows all disclosure flags below 'add' and 'rem' though when used each disclosure may apear at most in one of both).

CAUTION: Modifying contact flags is not allowed.

### Server response

The server answers with result code 2303 if a contact/holder with the given ID did not exist, 2304 if update was not possible because of contact's flags (client or server update prohibited flag).

On successful update of the contact/holder a result code of 1000 is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:iana:xml:ns:epp"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="urn:iana:xml:ns:epp␣epp.xsd">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>560C9BCA-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample contact/holder update response

### 5.4.5 EPP delete command

This command follows standard EPP - Contact mapping, rfc3733[5].

#### Client request

The client issues a contact delete request to delete a contact. The deletion is only possible if the contact has none of the delete prohibited flags and is not in use by any domain object.

```
<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>
         <contact:delete xmlns:contact="urn:ietf:params:xml:ns:contact-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:contact-1.0␣contact-1.0.xsd">
            <contact:id>C100</contact:id>
         </contact:delete>
      </delete>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample contact/holder delete request

All fields must be printable US-ASCII.

| | |
|---|---|
| **id** | ID of contact/holder to delete |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 2303 if a contact/holder with the given ID did not exist, 2304 if deletion was not possible because of contact's flags (client or server delete prohibited flag), 2305 if the contact is referenced by a domain object.

On successful deletion of the contact/holder a result code of 1000 is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:iana:xml:ns:epp"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="urn:iana:xml:ns:epp␣epp.xsd">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>560C9BCA-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample contact/holder delete response

## 5.5 Host-specific EPP commands and corresponding procedures

This section describes the EPP commands that apply to host objects, as specified in rfc3732[4].

DNS-LU allows zero or one IPv4 address and zero or one IPv6 address for each hostname ending in .lu. All other hosts may not have an IP address. For hosts that are nameservers to their own domain the IP address (v4, v6 or both) must be provided. Hostnames may not end in . and must all be fully qualified.

Thus a host with name 'ns1' is not permitted and must be expressed as 'ns1.domain.lu'.

IDN hostnames must be handled with their ACE encoded name.

### 5.5.1 EPP check command

This command follows standard EPP - Host mapping, rfc3732[4]. For DNS-LU IDN hosts must be manipulated using the ACE encoded host name.

#### Client request

The client issues a host check request to determine if host names are available for new use or if they are already in use.

```
<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>
         <host:check xmlns:host="urn:ietf:params:xml:ns:host-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:host-1.0␣host-1.0.xsd">
            <host:name>ns1.dns.lu</host:name>
            <host:name>ns2.dns.lu</host:name>
            <host:name>ns.mydns.org</host:name>
            <host:name>ns2.adomain.lu</host:name>
         </host:check>
      </check>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample host check request

All fields must be printable US-ASCII.

| | |
|---|---|
| **name** | (unbounded) Hostname to check |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 1000 on successfull processing of the request.

For each requested hostname the server indicates if the hostname is available for creating new hosts.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>
         <host:chkData xmlns:host="urn:ietf:params:xml:ns:host-1.0"
```

```
            xsi:schemaLocation="urn:ietf:params:xml:ns:host-1.0␣host-1.0.xsd">
         <host:cd>
            <host:name avail="0">ns1.dns.lu</host:name>
         </host:cd>
         <host:cd>
            <host:name avail="1">ns2.dns.lu</host:name>
         </host:cd>
         <host:cd>
            <host:name avail="1">ns.mydns.org</host:name>
         </host:cd>
         <host:cd>
            <host:name avail="1">ns2.adomain.lu</host:name>
         </host:cd>
      </host:chkData>
   </resData>
   <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>7DBEEDD0-DNSLU</svTRID>
   </trID>
</response>
</epp>
```

Sample host check response

### 5.5.2 EPP info command

This command follows standard EPP - Host mapping, rfc3732[4]. For DNS-LU IDN hosts must be manipulated using the ACE encoded host name.

#### Client request

The client issues a host info request to fetch a host's details.

```
<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>
         <host:info xmlns:host="urn:ietf:params:xml:ns:host-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:host-1.0␣host-1.0.xsd">
            <host:name>ns1.dns.lu</host:name>
         </host:info>
      </info>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample host info request

All fields must be printable US-ASCII.

| | |
|---|---|
| **name** | Name of host whose info to fetch |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 2303 if a host with the given ID did not exist.

On successful processing of the host info request a result code of 1000 is returned and a host:infData item provides detailled information about the host object.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>
         <host:infData xmlns:host="urn:ietf:params:xml:ns:host-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:host-1.0␣host-1.0.xsd">
            <host:name>ns1.dns.lu</host:name>
            <host:roid>N123-DNSLU</host:name>
            <host:status s="pendingUpdate"/>
            <host:addr ip="v4">158.64.101.101</host:addr>
            <host:addr ip="v6">3ffe:0501:0008:0000:0260:97ff:fe40:efab</host:addr>
            <host:clID>restena-id</host:crID>
            <host:crID>restena-id</host:crID>
            <host:crDate>2004-09-09T08:14:13.0Z</host:crDate>
            <host:upID>restena-id</host:upID>
            <host:upDate>2004-09-09T10:02:19.0Z</host:upDate>
         </host:infData>
      </resData>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>19CB8E95-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample host info response

All fields are printable US-ASCII.

| | |
|---|---|
| **name** | Host name |
| **roid** | Registry unique ID for the host |
| **status** | (unbounded) status flags as defined by EPP |
| **addr[@ip=v4]** | (optional) IPv4 address |
| **addr[@ip=v6]** | (optional) IPv6 address |
| **clID** | Registrar who owns the host object |
| **crID** | registrar who created host |
| **crDate** | creation date of host |
| **upID** | (optional) Registrar who did last update to host |
| **upDate** | (optional) Date of last update to host |
| **clTRID** | Client transaction ID |

### 5.5.3 EPP create command

This command follows standard EPP - Host mapping, rfc3732[4].

DNS-LU limitations on rfc3732[4] do exist. At most one IPv4 and one IPv6 address may be provided. IP addresses are allowed only for local (*.lu) hosts. Host name must be fully qualified and IDN host names must be ACE encoded.

#### Client request

The client issues a host create request to create a host.

```
<?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>
         <host:create xmlns:host="urn:ietf:params:xml:ns:host-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:host-1.0␣host-1.0.xsd">
            <host:name>ns.domain.lu</host:name>
            <host:addr ip="v4">158.64.229.2</host:addr>
            <host:addr ip="v6">1080:0:0:0:8:800:200C:417A</host:addr>
```

```
         </host:create>
      </create>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample host create request

All fields must be printable US-ASCII.

| | |
|---|---|
| **name** | Fully qualified host name |
| **addr[@ip=v4]** | (optional) IPv4 address |
| **addr[@ip=v6]** | (optional) IPv6 address |
| **clTRID** | Client transaction ID |

**Server response**

The server answers with result code 2302 if a host with the given name already existed.

On successful creation of the new host a result code of 1000 is returned and a host:creData item provides limited information about the creation.

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <resData>
         <host:creData xmlns:host="urn:ietf:params:xml:ns:host-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:host-1.0␣host-1.0.xsd">
            <host:name>ns.domain.lu</host:name>
            <host:crDate>2005-10-04T14:36:21Z</host:crDate>
         </host:creData>
      </resData>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>61E0CEFC-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample host create response

All fields are printable US-ASCII.

| | |
|---|---|
| **name** | Host name |
| **crDate** | UTC creation data of the object |
| **clTRID** | Client transaction ID |
| **svTRID** | Server transaction ID |

### 5.5.4 EPP update command

This command follows standard EPP - Host mapping, rfc3732[4].

DNS-LU limitations on rfc3732[4] do exist. A host may have at most one IPv4 and one IPv6 address. IP addresses are allowed only for local (*.lu) hosts. Host name must be fully qualified and IDN host names must be ACE encoded.

When replacing IP addresses, the old one must be removed and the new one added. Host name can only be changed from *.lu to *.lu or non-lu host name to non-lu host name, and may not be replaced by itself.

For any host linked to it's active parent domain, updates to IP addresses or hostname will cause the complete update to be pending.

Status updates (clientUpdateProhibited, clientDeleteProhibited) are not avilable.

### Client request

The client issues a host update request to modify a host's details.

```
<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>
         <host:update xmlns:host="urn:ietf:params:xml:ns:host-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:host-1.0␣host-1.0.xsd">
            <host:name>ns.domain.lu</host:name>
            <host:add>
               <host:addr ip="v4">123.45.67.8</host:addr>
               <host:addr ip="v6">1080:0:0:0:8:800:200C:417B</host:addr>
            </host:add>
            <host:rem>
               <host:addr ip="v4">123.45.67.2</host:addr>
               <host:addr ip="v6">1080:0:0:0:8:800:200C:417A</host:addr>
            </host:rem>
            <host:chg>
               <host:name>ns1.domain.lu</host:name>
            </host:chg>
         </host:update>
      </update>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample host create request

All fields must be printable US-ASCII.

| | |
|---|---|
| **name** | Fully qualified host name |
| **add/addr[@ip=v4]** | (optional) New IPv4 address to add to host |
| **add/addr[@ip=v6]** | (optional) New IPv6 address to add to host |
| **rem/addr[@ip=v4]** | (optional) Old IPv4 address to remove from host |
| **rem/addr[@ip=v6]** | (optional) Old IPv6 address to remove from host |
| **chg/name** | New fully qualified host name |
| **clTRID** | Client transaction ID |

CAUTION: Modifying host flags is not allowed.

### Server response

The server answers with result code 2303 if a host with the given name did not exist, 2304 if update was not possible because of host's flags (client or server update prohibited flag) and 2302 if host's name is being changed an a how with the new name already exists.

On successful update of the host a result code of 1000 is returned. In case of pending update result code 1001 is returned, at a later time a poll message will indicate processing of this pending operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>560C9BCA-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample host update response

### 5.5.5 EPP delete command

This command follows standard EPP - Host mapping, rfc3732[4]. For DNS-LU IDN hosts must be manipulated using the ACE encoded host name.

#### Client request

The client issues a host info request to fetch a host's details.

```
<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>
         <host:delete xmlns:host="urn:ietf:params:xml:ns:host-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:host-1.0␣host-1.0.xsd">
            <host:name>ns1.dns.lu</host:name>
         </host:delete>
      </delete>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample host delete request

All fields must be printable US-ASCII.

| | |
|---|---|
| **name** | Name of host whose to delete |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 2303 if a host with the given name did not exist, 2304 if deletion was not possible because of host's flags (client or server delete prohibited flag), 2305 if the host is referenced by a domain object.

On successful deletion of the host a result code of 1000 is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>560C9BCA-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample host delete response

## 5.6 Domain-specific EPP commands and corresponding procedures

This section describes the EPP commands that apply to domain objects, as specified in rfc3731[3].

Domains are always referenced by their US-ASCII name (ACE encoded name for IDN domains) and must have at least two working nameservers associated to be marked active. In domain creation commands and transfer/trade commands both IDN and ACE encoded domain name must be provided and conversion must succeed and match in both directions: ACE(IDN) == ACE, IDN(ACE) == IDN.

Possible domain names can be grouped in 4 categories:

- names requiring special approval for registration (e.g. village names). Registration of these must be validated by human intervention.
- blacklisted domain names can never be registered (either because they are reserved for future use or they are against naming policy)
- out-of-range domain names can never be registered (too short names, too long names, invalid characters)
- remaining names that get auto-registered (exported after nameserver validation)

For transfer, transfer-trade, transfer-restore and trade requests, the optional execution date must be in the future (if provided). Omitting this date is interpreted as 'as soon as possible'. Note that this date only takes effect once the operation has been accepted as requested in an email to the current administrative contact.

### 5.6.1 EPP check command

This command follows standard EPP - Domain name mapping, rfc3731[3]. No DNS-LU extensions or restrictions.

For each domain to be tested information about availability or special cases is returned. All domains that are either pendingCreate, pendingDelete, pendingUpdate or ok are described as already registered (without further details, these can be obtained by domain info command). Domain names that fall into category of village names will have a statement about this in the response reason, if available. Domain names that are not valid or not acceptable will be marked as such.

When checking for IDN domains the ACE encoded domain name has to be provided.

#### Client request

The client issues a domain check request to determine if domain names are available for registration or if they are already in use.

```
<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>
         <domain:check xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>luxembourg.lu</domain:name>
            <domain:name>xn--lyce-dpa.lu</domain:name>
            <domain:name>restena.lu</domain:name>
```

```
            <domain:name>some-domain.lu</domain:name>
         </domain:check>
      </check>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample domain check request

All fields must be printable US-ASCII.

| | |
|---|---|
| **name** | (unbounded) Domain name to check |
| **clTRID** | Client transaction ID |

**Server response**

The server answers with result code 1000 on successful process of the request.

For each requested domain name the server indicates if the domain name is available for creating new domains and whether some limitations apply.

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <resData>
         <domain:chkData xmlns:domain="urn:ietf:params:xml:ns:host-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:cd>
               <domain:name avail="0">luxembourg.lu</domain:name>
               <domain:reason lang="en">Village name, in use</domain:reason>
            </domain:cd>
            <domain:cd>
               <domain:name avail="0">xn--lyce-dpa.lu</domain:name>
               <domain:reason lang="en">IDN domains are not allowed</domain:reason>
            </domain:cd>
            <domain:cd>
               <domain:name avail="0">restena.lu</domain:name>
               <domain:reason lang="en">In use</domain:reason>
            </domain:cd>
            <domain:cd>
               <domain:name avail="1">some-domain.lu</domain:name>
            </domain:cd>
         </domain:chkData>
      </resData>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>7919DACB-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain check response

### 5.6.2 EPP info command

This command follows standard EPP - Domain name mapping, rfc3731[3]. DNS-LU extension adds extra status flags as well as explicit IDN support.

This command allows to retrieve information about a domain object. The information returned depends on who requests the information. Registrar sponsoring a domain will get all information available except pendingTransfer status flag.

Nonsponsoring registrars will get a subset of the information like wether the domain is active or not, subset of status flags, id of sponsoring registrar.

### Client request

The client issues a domain info request to fetch a domain's details.

```
<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>
         <domain:info xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name hosts="all">xn--lyce-dpa.lu</domain:name>
         </domain:info>
      </info>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample domain info request

All fields must be printable US-ASCII.

| | |
|---|---|
| **name** | Name of host whose info to fetch |
| **clTRID** | Client transaction ID |

### Server response

The server answers with result code 2303 if a domain with the given name did not exist.

On successful processing of the domain info request a result code of 1000 is returned and a domain:infData item provides detailled information about the domain object.

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <resData>
         <domain:infData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>xn--lyce-dpa.lu</domain:name>
            <domain:roid>D0-DNSLU</domain:roid>
            <domain:status s="pendingCreate"/>
            <domain:status s="inactive"/>
            <domain:registrant>H100</domain:registrant>
            <domain:contact type="admin">C100</domain:contact>
            <domain:contact type="tech">C100</domain:contact>
            <domain:ns>
               <domain:hostObj>ns.restena.lu</domain:hostObj>
            </domain:ns>
            <domain:host>ns1.xn--lyce-dpa.lu</domain:host>
            <domain:host>ns6.xn--lyce-dpa.lu</domain:host>
            <domain:clID>restena-id</domain:clID>
            <domain:crID>restena-id</domain:crID>
            <domain:crDate>2005-10-03T17:22:31Z</domain:crDate>
            <domain:upID>restena-id</domain:upID>
            <domain:upDate>2006-06-27T11:10:46Z</domain:upDate>
            <domain:exDate>2006-10-03T17:22:31Z</domain:exDate>
         </domain:infData>
      </resData>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/200/10/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:infData>
                  <dnslu:domain>
                     <dnslu:idn>lycée.lu</dnslu:idn>
```

```
                    <dnslu:status>serverTradeProhibited</dnslu:status>
                    <dnslu:crReqID>restena-id</dnslu:crReqID>
                    <dnslu:crReqDate>2005-10-03T11:37:22Z</dnslu:crReqDate>
                    <dnslu:delReqDate>2006-07-03T11:12:12Z</dnslu:delReqDate>
                    <dnslu:delDate>2006-07-21T17:37:54Z</dnslu:delReqDate>
                 </dnslu:domain>
              </dnslu:infData>
           </dnslu:resData>
        </dnslu:ext>
     </extension>
     <trID>
        <clTRID>ABC-12345</clTRID>
        <svTRID>42C1B707-DNSLU</svTRID>
     </trID>
  </response>
</epp>
```

Sample domain info response on sponsored domain

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **roid** | Registry unique ID for the holder |
| **status** | (unbounded) status flags as defined by EPP |
| **registrant** | ID of domain holder |
| **contact[@type=admin]** | ID of Administrative contact |
| **contact[@type=admin]** | ID of Technical contact |
| **hostObj** | (unbounded) list of nameserver for the domain |
| **host** | (unbounded) List of host objects below this domain |
| **clID** | Registrar who sponsors the domain |
| **crID** | Registrar who created domain |
| **crDate** | Creation date of domain |
| **upID** | (optional) Registrar who did last update to domain |
| **upDate** | (optional) Date of last update to domain |
| **exDate** | (optional) Domain expiration date |
| **dnslu:idn** | (extension, optional) IDN name for IDN domains, UTF-8 |
| **dnslu:status** | (extension, unbounded) extra status flags as defined by DNS-LU |
| **dnslu:crReqID** | (extension, optional) Registrar who requested domain registration |
| **dnslu:crReqDate** | (extension, optional) Date of domain registration request |
| **dnslu:delReqDate** | (extension, optional) Date of deletion request (setDate) |
| **dnslu:delDate** | (extension, optional) Date deletion execution |
| **clTRID** | Client transaction ID |

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <resData>
         <domain:infData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>xn--lyce-dpa.lu</domain:name>
            <domain:roid>D0-DNSLU</domain:roid>
            <domain:status s="inactive"/>
            <domain:clID>restena-id</domain:clID>
            <domain:host>ns3.xn--lyce-dpa.lu</domain:host>
         </domain:infData>
      </resData>
      <extension>
```

```
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/200/10/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:infData>
                  <dnslu:domain>
                     <dnslu:idn>lycée.lu</dnslu:idn>
                  </dnslu:domain>
               </dnslu:infData>
            </dnslu:resData>
         </dnslu:ext>
      </extension>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>79E116D4-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain info response on non-sponsored domain

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **roid** | Registry unique ID for the holder |
| **status** | (unbounded) subset of status flags defined by EPP |
| **host** | (unbounded) List of host objects below this domain |
| **clID** | Registrar who sponsors the domain |
| **crID** | Registrar who created domain |
| **crDate** | Creation date of domain |
| **exDate** | (optional) Domain expiration date |
| **dnslu:idn** | (extension, optional) Show/hide name in WHOIS |
| **dnslu:status** | (extension, unbounded) subset of extra status flags as defined by DNS-LU |
| **clTRID** | Client transaction ID |

**Note**: If a transfer of the querried domain is pending, then contact and host details that will be in effect after successful transfer are shown as the current contacts and hosts would be shown for the sponsoring registrar.

### 5.6.3 EPP create command

This command extends standard EPP - Domain name mapping, rfc3731[3]. DNS-LU extension adds status flag setting at creation as well as explicit IDN support.

DNS-LU limitations on rfc3731[3] do exist as well. The `<domain:authInfo>` tag is ignored (but still has to be present to fullfill Schema requirements). The optional `<domain:period>` tag is not accepted by DNS-LU as renewal happens automatically. DNS-LU requires exactly one administrative and one technical contact for a domain.

#### Client request

The client issues a domain create request to register a new domain.

**Note**: By default a domain will be created as active and needs two nameservers. To reserve a domain name, the inactive status flag must be added in the extension part.

```
<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>
         <domain:create xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
```

```
            xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0 domain-1.0.xsd">
         <domain:name>xn--lyce-dpa.lu</domain:name>
         <domain:ns>
            <domain:hostObj>ns1.restena.lu</domain:hostObj>
            <domain:hostObj>ns2.restena.lu</domain:hostObj>
         </domain:ns>
         <domain:registrant>H_rest</domain:registrant>
         <domain:contact type="admin">CA_rest</domain:contact>
         <domain:contact type="tech">CT_rest</domain:contact>
         <domain:authInfo>
            <domain:pw>dummy</domain:pw>
         </domain:authInfo>
      </domain:create>
   </create>
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0 dnslu-1.0.xsd">
         <dnslu:create>
            <dnslu:domain>
               <dnslu:idn>lycée.lu</dnslu:idn>
               <dnslu:status s="inactive"/>
               <dnslu:status s="clientTradeProhibited"/>
            </dnslu:domain>
         </dnslu:create>
      </dnslu:ext>
   </extension>
   <clTRID>ABC-12345</clTRID>
</command>
</epp>
```

Sample domain create request

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **hostObj** | (unbounded) list of nameserver for the domain |
| **registrant** | ID of domain holder |
| **contact[@type=admin]** | ID of Administrative contact |
| **contact[@type=tech]** | ID of Technical contact |
| **authInfo** | (ignored) |
| **dnslu:idn** | (extension) Must be IDN equivalent of `xn--` name, UTF-8<br>Not allowed for non-IDN domain names |
| **dnslu:status** | (extension, unbounded) status flags (DNS-LU and EPP status flags) |
| **clTRID** | Client transaction ID |

### Server response

The server answers with result code 2302 if a domain with the given name already existed, 2306 if the domain name is black-listed.

On successful creation request of the new domain a result code of 1001 is returned. A poll message will be issued when nameserver checks and domain name/holder validation has been performed as needed. In case of failure of nameservers the domain will be created as reserved.

**Note**: The creData element in the response contains the domain name and the creation request date. This date is NOT the creation date as creation may be delayed for a longer time depending on domain name or current policy. Expiration date is not indicate in creData as this information is undefined at this point in time. For these dates the matching poll message should be checked. Domain creation status may also be checked using domain info 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">
```

```
   <response>
      <result code="1001">
         <msg>Command completed successfully; action pending</msg>
      </result>
      <resData>
         <domain:creData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0" xsi:schemaLocation="urn:ietf:params:xml:ns:d
            <domain:name>xn--lyce-dpa.lu</domain:name>
            <domain:crDate>2006-06-15T15:21:45.000Z</domain:crDate>
         </domain:creData>
      </resData>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>6C0B58E0-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain create response

### 5.6.4 EPP update command

This command extends standard EPP - Domain name mapping, rfc3731[3]. DNS-LU extension adds extra status flag.

DNS-LU limitation on rfc3731[3] do exist as well. The optional `<domain:chg>` tag is not accepted. (To change holder issue a trade command) Changing contacts is done by removing old contact and adding a new one in same command. A domain must always have exactly one administrative and one technical contact.

#### Client request

The client issues a domain update request to update a domain's details.

**Note**: Some flags may be disabled for setting by registrars as clientUpdateProhibited, clientDeleteProhibited, clientTradeProhibited and clientTransferProhibited at time of writing.

**Note**: Adding inactive status flag makes domain reserved, removing it makes domain active.

```
<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>
         <domain:update xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>xn--lyce-dpa.lu</domain:name>
            <domain:add>
               <domain:ns>
                  <domain:hostObj>ns1.restena.lu</domain:hostObj>
                  <domain:hostObj>ns2.restena.lu</domain:hostObj>
                  <domain:hostObj>ns3.restena.lu</domain:hostObj>
               </domain:ns>
               <domain:contact type="admin">C100</domain:contact>
               <domain:contact type="tech">C100</domain:contact>
               <domain:status s="clientUpdateProhibited"/>
            </domain:add>
            <domain:rem>
               <domain:ns>
                  <domain:hostObj>ns1.restena.lu</domain:hostObj>
                  <domain:hostObj>ns2.restena.lu</domain:hostObj>
                  <domain:hostObj>ns3.restena.lu</domain:hostObj>
               </domain:ns>
               <domain:contact type="admin">C100</domain:contact>
               <domain:contact type="tech">C100</domain:contact>
               <domain:status s="clientDeleteProhibited"/>
            </domain:rem>
         </domain:update>
      </update>
      <extension>
```

```
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:update>
               <dnslu:domain>
                  <dnslu:add>
                     <dnslu:status s="clientTradeProhibited"/>
                  </dnslu:add>
                  <dnslu:rem>
                     <dnslu:status s="clientTransferProhibited"/>
                  </dnslu:rem>
               </dnslu:domain>
            </dnslu:update>
         </dnslu:ext>
      </extension>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample domain update request

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **ns/hostObj** | (unbounded) list of nameserver for the domain |
| **contact[@type=admin]** | Add/remove ID of Administrative contact |
| **contact[@type=tech]** | Add/remove ID of Technical contact |
| **status** | (unbounded) add/remove EPP status flags |
| **authInfo** | (not allowed) |
| **dnslu:status** | (extension, unbounded) add/remove extra status flags as defined by DNS-LU |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 2303 if a domain with the given name did not exist, 2304 if update was not possible because of domain's flags (client or server update prohibited flag, pending states). If nameserver count is out of range error 2004 is returned.

On successful update of the domain a result code of 1000 is returned. In case of pending update result code 1001 is returned, at a later time a poll message will indicate processing of this pending operation. Update is pending whenever the domain should be active after update and previously was reserved or nameservers are added/removed by the update.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:iana:xml:ns:epp"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="urn:iana:xml:ns:epp␣epp.xsd">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>1788F01D-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain update response

### 5.6.5 EPP delete immediate command

This command extends standard EPP - Domain name mapping, rfc3731[3]. DNS-LU extension add two additionnal delete operations: setDate and cancel. The standard EPP delete command

has operation name `immediate` which can be optionally specified. `setDate` operation (described in section 5.6.6) schedules a domain for deletion at a given date and `cancel` (described in section 5.6.7) removes a scheduled deletion.

### Client request

The client issues a domain delete request to delete a domain.

**Note**: On deletion the domain goes into pendingDelete state (also known as quarantine state) from where it can be restored during a period of one month (as of writing). In this state the domain may not be modified, traded or newly registered, but will not be exported to DNS servers or WHOIS either.

```
<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>
         <domain:delete xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>domain.lu</domain:name>
         </domain:delete>
      </delete>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:delete>
               <dnslu:domain>
                  <dnslu:op>immediate</dnslu:op>
               </dnslu:domain>
            </dnslu:delete>
         </dnslu:ext>
      </extension>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample domain delete request

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **dnslu:op** | (extension, optional) If specified must be 'immediate' |
| **clTRID** | Client transaction ID |

**Note**: The extension is optional for delete immediate command as this is the standard behavior defined by EPP.

### Server response

The server answers with result code 2303 if a domain with the given name did not exist, 2304 if update was not possible because of domain's flags (client or server delete prohibited flag, some pending states)

On successful processing of the domain delete request a result code of 1000 is returned and the domain is changed to state pendingDelete (quarantine).

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
```

```
      </result>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>3C265C89-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain delete response

### 5.6.6 EPP delete setDate command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. The `setDate` delete command schedules a domain for deletion at a given date. Scheduler will put domain into pendingDelete once the scheduled date is reached. This scheduling can be cancelled with `cancel` operation (described in section 5.6.7)

#### Client request

The client issues a domain delete setDate request to schedule deletition of a domain.

**Note**: Scheduled deletion has no influence on normal operation of the scheduled domain. Once the scheduled time is reached the domain will be deleted (as if a domain delete immediate command was issued at that moment).

```
<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>
         <domain:delete xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>domain.lu</domain:name>
         </domain:delete>
      </delete>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:delete>
               <dnslu:domain>
                  <dnslu:op>setDate</dnslu:op>
                  <dnslu:delDate>2005-11-08T00:00:00Z</dnslu:delDate>
               </dnslu:domain>
            </dnslu:delete>
         </dnslu:ext>
      </extension>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample domain delete setDate request

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **dnslu:op** | (extension) Must be 'setDate' |
| **dnslu:delDate** | (extension) Date at which deletion should happen |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 2303 if a domain with the given name did not exist, 2304 if delete setDate was not possible because of domain's flags (client or server delete prohibited flag,

some pending states). If the deletion date is either in the past or too far in the future a result code of 2004 will be returned.

On successful processing of the domain delete setDate request a result code of 1000 is returned and the domain is scheduled for deletition at given date.

**Note**: The deletion operation may not be executed at exact specified time. If the date specified is earlier than expiration date the domain will not be renewed right before deletion. To prevent race conditions, always schedule operations at least 24 hours in advance.

```
<?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">
   <response>
      <result code="1000">
         <msg>Command completed successfully</msg>
      </result>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>4BA08CD4-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain delete setDate response

### 5.6.7 EPP delete cancel command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. The `cancel` delete command removes a scheduled domain deletition. This command undoes what `setDate` operation (described in section 5.6.6) did, as long as domain is not pendingDelete.

Domain object deletions that have been requested using the delete immediate command described in section 5.6.5 or which were already deleted by the scheduler cannot be cancelled, since those requests are executed immediately by the EPP server [2].

#### Client request

The client issues a domain delete cancel request to unschedule deletion of a domain.

```
<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>
         <domain:delete xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>domain.lu</domain:name>
         </domain:delete>
      </delete>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:delete>
               <dnslu:domain>
                  <dnslu:op>cancel</dnslu:op>
               </dnslu:domain>
            </dnslu:delete>
         </dnslu:ext>
      </extension>
      <clTRID>ABC-12345</clTRID>
   </command>
```

[2] If a domain object is in state pendingDelete, the deletion cannot be cancelled, but the domain object can be restored using the EPP domain restore command (cf. section 5.6.8).

```
</epp>
```

Sample domain delete cancel request

All fields are printable US-ASCII

| | |
|---|---|
| **name** | Domain name |
| **dnslu:op** | (extension) Must be 'cancel' |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 2303 if a domain with the given name did not exist, 2304 if update was not possible because of domain's flags (client or server delete prohibited flag, non-scheduled deletion).

On successful processing of the domain delete cancel request a result code of 1000 is returned and the domain deletion is unscheduled.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>1788F01D-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain delete cancel response

### 5.6.8 EPP restore command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. The `restore` command moves a pendingDelete domain back to active preserving the old state of the domain.

When a domain is restored, a new registration year begins. The period while the domain is pendingDelete is called quarantine period and lasts during a configured period starting from the date when the domain was deleted. (1 month at the time of writing)

#### Client request

The client issues a domain restore request to recover a domain that is in pendingDelete state.

```
<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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:restore>
               <dnslu:domain>
                  <dnslu:name>domain.lu</dnslu:name>
               </dnslu:domain>
            </dnslu:restore>
```

```
            <dnslu:clTRID>1288301D-DNSLU</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain restore request

All fields must be printable US-ASCII.

| | |
|---|---|
| **name** | Name of host whose info to fetch |
| **clTRID** | Client transaction ID |

#### Server response

The server answers with result code 2303 if a domain with the given name did not exist, 2304 if restore was not possible because domain was not pendingDelete.

On successful processing of the domain info request a result code of 1000 is returned and the domain is restored.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>1788F01D-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain restore response

### 5.6.9 EPP transfer request command

This command is standard EPP with DNS-LU extensions - Domain name mapping, rfc3731[3]. The `request` transfer command is used to change the registrar sponsoring a domain.

This command is nearly equal to trade request command, with difference that the domain name is provided using the standard EPP domain mapping. As for trade request, this operation is equivalent to the domain create command, but requires the domain to exist in another registrar's view and be linked to a holder describing the same entity. Until the transfer is executed by the scheduler the transfer request may be cancelled using `cancel` transfer command (described in section 5.6.11).

#### Client request

The client issues a domain transfer request to obtain sponsorship for a domain.

For each contact, when permitted by server configuration a copy of current data can be requested with a new contact ID prefixed by symbol 'ª' (Unicode 00AA, "FEMININE ORDINAL INDICATOR"). A contact ID 'CTID' requests use of existing contact with ID 'CTID'; a contact ID 'ªCTID' requests use of non-existing contact with ID 'CTID'. Please read section 3.2.21 about copy of contacts for usage restrictions.

**Note**: By default a domain will be transfered as active and needs two nameservers. To reserve a domain name, the inactive status flag must be added in the extension part.

```
<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">
         <domain:transfer xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>xn--caf-dma.lu</domain:name>
         </domain:transfer>
      </transfer>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:transfer>
               <dnslu:domain>
                  <dnslu:ns>
                     <dnslu:hostObj>ns1.restena.lu</dnslu:hostObj>
                     <dnslu:hostObj>ns2.restena.lu</dnslu:hostObj>
                     <dnslu:hostObj>ns3.restena.lu</dnslu:hostObj>
                  </dnslu:ns>
                  <dnslu:registrant>H100</dnslu:registrant>
                  <dnslu:contact type="admin">C100</dnslu:contact>
                  <dnslu:contact type="tech">C100</dnslu:contact>
                  <dnslu:status s="clientHold"/>
                  <dnslu:idn>café.lu</dnslu:idn>
                  <dnslu:trDate>2004-06-30T12:00:00Z</dnslu:trDate>
               </dnslu:domain>
            </dnslu:transfer>
         </dnslu:ext>
      </extension>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample domain transfer request

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **hostObj** | (unbounded) list of nameserver for the domain |
| **registrant** | ID of domain holder |
| **contact[@type=admin]** | ID of Administrative contact |
| **contact[@type=tech]** | ID of Technical contact |
| **status** | (extension, unbounded) status flags (DNS-LU and EPP status flags) |
| **idn** | (extension) Must be IDN equivalent of `xn--` name, UTF-8<br>Not allowed for non-IDN domain names |
| **trDate** | (optional) Explicit date at which trade should be executed |
| **clTRID** | Client transaction ID |

### Server response

The server answers with result code 2106 if a domain with the given name is sponsored by the registrar itself. Code 2300 is returned if the domain is already pending trade, transfer, transfer-trade or transfer-restore.

On successful transfer request of the domain a result code of 1001 is returned. A poll message will be issued when transfer is aborted/executed. Current status of the transfer can be checked with transfer query command described later on. The response contains a trnData element that gives information on the requested transfer operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0␣epp-1.0.xsd">
```

```
   <response>
      <result code="1001">
         <msg>Command completed successfully; action pending</msg>
      </result>
      <resData>
         <domain:trnData xmlns="urn:ietf:params:xml:ns:domain-1.0"
              xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <dnslu:name>xn--caf-dma.lu</dnslu:name>
            <dnslu:trStatus>pending</dnslu:trStatus>
            <dnslu:reID>restena-id</dnslu:reDate>
            <dnslu:reDate>2004-09-08T11:39:41Z</dnslu:reDate>
            <dnslu:acDate>2004-09-15T11:39:41Z</dnslu:reDate>
         </domain:trnData>
      </resData>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:trnData>
                  <dnslu:domain>
                     <dnslu:trDate>2004-09-18T10:00:00Z</dnslu:trDate>
                  </dnslu:domain>
               </dnslu:trnData>
            </dnslu:resData>
         </dnslu:ext>
      </extension>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>30C7B5B-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain transfer request response

All fields are printable US-ASCII

| | |
|---|---|
| **name** | Domain name |
| **trStatus** | Transfer status (see EPP transfer status for possible values) |
| **reID** | ID of registrar asking for transfer |
| **reDate** | Date when transfer was requested |
| **acDate** | Date when administrative contact approved/rejected transfer |
| **trDate** | Date when transfer should happen (is possible) |
| **clTRID** | Client transaction ID |
| **svTRID** | Server transaction ID |

### 5.6.10 EPP transfer query command

This command is standard EPP - Domain name mapping, rfc3731[3]. It provides a query operation that allows an EPP client to determine the real-time status of ongoing transfer requests (that have been requested using the EPP domain transfer request command described in section 5.6.9).

#### Client request

The client issues a domain transfer query request check transfer status for a domain.

```
<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="query">
         <domain:transfer xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
                          xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>domain.lu</domain:name>
```

```
         </domain:transfer>
      </transfer>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample domain transfer query request

**Server response**

The server answers with result code 2303 if no domain with given name exists, 2301 if domain is not being transfered.

On successful querrying of the transfer status a result code of 1000 is returned and transfer information is returned as in transfer request response.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>
         <domain:trnData xmlns="urn:ietf:params:xml:ns:domain-1.0"
                         xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <dnslu:name>domain.lu</dnslu:name>
            <dnslu:trStatus>pending</dnslu:trStatus>
            <dnslu:reID>restena-id</dnslu:reDate>
            <dnslu:reDate>2004-09-08T11:39:41Z</dnslu:reDate>
            <dnslu:acDate>2004-09-15T11:39:41Z</dnslu:reDate>
         </domain:trnData>
      </resData>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:trnData>
                  <dnslu:domain>
                     <dnslu:trDate>2004-09-18T10:00:00Z</dnslu:trDate>
                  </dnslu:domain>
               </dnslu:trnData>
            </dnslu:resData>
         </dnslu:ext>
      </extension>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>30C7B5B-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain transfer query response

See transfer request response for details.

### 5.6.11 EPP transfer cancel command

This command is standard EPP - Domain name mapping, rfc3731[3]. It provides a cancel operation that allows an EPP client to cancel a domain transfer (that has been requested using the EPP domain transfer request command described in section 5.6.9).

**Client request**

The client issues a domain transfer cancel request cancel a domain transfer (**Note**: cancelled/aborted transfers get charged).

```
<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="cancel">
         <domain:transfer xmlns:domain="urn:ietf:params:xml:ns:domain-1.0"
                          xsi:schemaLocation="urn:ietf:params:xml:ns:domain-1.0␣domain-1.0.xsd">
            <domain:name>domain.lu</domain:name>
         </domain:transfer>
      </transfer>
      <clTRID>ABC-12345</clTRID>
   </command>
</epp>
```

Sample domain transfer cancel request

#### Server response

The server answers with result code 2303 if no domain with given name exists, 2301 if cancelling transfer is not possible because domain is not being transferred.

On successful cancelling of the transfer a result code of 1000 is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>9478978-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain transfer cancel response

### 5.6.12 EPP trade request command

This command is an extension to standard EPP – Domain name mapping, rfc3731[3]. The `request` trade command is used to change the holder of a domain[3]. The operation is equivalent to the domain create command, but requires the domain to exist in registrar's view and be linked to another holder than the specified new one. Until the trade is executed by the scheduler the trade request may be cancelled using `cancel` trade command (described in section 5.6.14).

#### Client request

The client issues a domain trade request to change holder of a domain.

**Note**: By default a domain will be created as active and needs two nameservers. To reserve a domain name, the inactive status flag must be added in the extension part.

[3]EPP specifies that it is possible to update the holder reference of a domain object with the domain update command, but a specific trade command makes it possible that other attributes of a domain object can still be updated while the domain is in trade (the trade procedure can last several days). This is made possible by the fact that the domain object is not switched to some pendingTrade state while it is in trade. Keep also in mind that, by policy, for a change of holder a transfer(trade) command must be used.

```
<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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:trade op="request">
               <dnslu:domain>
                  <dnslu:name>xn--caf-dma.lu</dnslu:name>
                  <dnslu:ns>
                     <dnslu:hostObj>ns1.restena.lu</dnslu:hostObj>
                     <dnslu:hostObj>ns2.restena.lu</dnslu:hostObj>
                     <dnslu:hostObj>ns3.restena.lu</dnslu:hostObj>
                  </dnslu:ns>
                  <dnslu:registrant>H100</dnslu:registrant>
                  <dnslu:contact type="admin">C100</dnslu:contact>
                  <dnslu:contact type="tech">C100</dnslu:contact>
                  <dnslu:status s="clientHold"/>
                  <dnslu:idn>café.lu</dnslu:idn>
                  <dnslu:trDate>2004-06-30T12:00:00Z</dnslu:trDate>
               </dnslu:domain>
            </dnslu:trade>
            <dnslu:clTRID>ABC-12345</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain trade request

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **hostObj** | (unbounded) list of nameserver for the domain |
| **registrant** | ID of domain holder |
| **contact[@type=admin]** | ID of Administrative contact |
| **contact[@type=tech]** | ID of Technical contact |
| **status** | (extension, unbounded) status flags (DNS-LU and EPP status flags) |
| **idn** | (extension) Must be IDN equivalent of `xn--` name, UTF-8<br>Not allowed for non-IDN domain names |
| **trDate** | (optional) Explicit date at which trade should be executed |
| **clTRID** | Client transaction ID |

### Server response

The server answers with result code 2303 if a domain with the given name does not exist in the registrar's view (either does not exist at all or sponsored by some other registrar). Code 2300 is returned if the domain is already pending trade, transfer, transfer-trade or transfer-restore.

On successful trade request of the domain a result code of 1001 is returned. A poll message will be issued when trade is aborted/executed. Current status of the trade can be checked with trade query command described later on. The response contains a traData element that gives information on the requested trade operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0␣epp-1.0.xsd">
   <response>
      <result code="1001">
         <msg>Command completed successfully; action pending</msg>
      </result>
      <extension>
```

```
        <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                   xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
           <dnslu:resData>
              <dnslu:traData>
                 <dnslu:domain>
                    <dnslu:name>xn--caf-dma.lu</dnslu:name>
                    <dnslu:trStatus>pending</dnslu:trStatus>
                    <dnslu:reID>restena-id</dnslu:reDate>
                    <dnslu:reDate>2004-09-08T11:39:41Z</dnslu:reDate>
                    <dnslu:acDate>2004-09-15T11:39:41Z</dnslu:acDate>
                    <dnslu:trDate>2004-09-18T10:00:00Z</dnslu:trDate>
                 </dnslu:domain>
              </dnslu:traData>
           </dnslu:resData>
        </dnslu:ext>
     </extension>
     <trID>
        <clTRID>ABC-12345</clTRID>
        <svTRID>30C7B5B-DNSLU</svTRID>
     </trID>
  </response>
</epp>
```

Sample domain trade request response

All fields are printable US-ASCII

| | |
|---|---|
| **name** | Domain name |
| **trStatus** | Trade status (see EPP transfer status for possible values) |
| **reID** | ID of registrar asking for trade |
| **reDate** | Date when trade was requested |
| **acDate** | Date when administrative contact approved/rejected trade |
| **trDate** | Date when trade should happen (is possible) |
| **clTRID** | Client transaction ID |
| **svTRID** | Server transaction ID |

### 5.6.13 EPP trade query command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. It provides a query operation that allows an EPP client to determine the real-time status of ongoing trade requests (that have been requested using the EPP domain trade request command described in section 5.6.12).

#### Client request

The client issues a domain trade query request check trade status for a domain.

```
<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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:trade op="query">
               <dnslu:domain>
                  <dnslu:name>domain.lu</dnslu:name>
               </dnslu:domain>
            </dnslu:trade>
            <dnslu:clTRID>ABC-12345</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain trade query request

**Server response**

The server answers with result code 2303 if no domain with given name exists, 2301 if domain is not being traded.

On successful querrying of the trade status a result code of 1000 is returned and trade information is returned as in trade request response.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:traData>
                  <dnslu:domain>
                     <dnslu:name>domain.lu</dnslu:name>
                     <dnslu:trStatus>pending</dnslu:trStatus>
                     <dnslu:reID>restena-id</dnslu:reDate>
                     <dnslu:reDate>2004-09-08T11:39:41Z</dnslu:reDate>
                     <dnslu:acDate>2004-09-15T11:39:41Z</dnslu:reDate>
                     <dnslu:trDate>2004-09-18T10:00:00Z</dnslu:trDate>
                  </dnslu:domain>
               </dnslu:traData>
            </dnslu:resData>
         </dnslu:ext>
      </extension>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>30C7B5B-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain trade query response

See trade request response for details.

### 5.6.14 EPP trade cancel command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. It provides a cancel operation that allows an EPP client to cancel a domain trade (that has been requested using the EPP domain trade request command described in section 5.6.12).

**Client request**

The client issues a domain trade cancel request cancel a domain trade (**Note**: cancelled/aborted trades get charged).

```
<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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
```

```
            <dnslu:trade op="cancel">
               <dnslu:domain>
                  <dnslu:name>domain.lu</dnslu:name>
               </dnslu:domain>
            </dnslu:trade>
            <dnslu:clTRID>ABC-12345</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain trade cancel request

#### Server response

The server answers with result code 2303 if no domain with given name exists, 2301 if cancelling trade is not possible because domain is not being traded.

On successful cancelling of the trade a result code of 1000 is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>9478978-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain trade cancel response

### 5.6.15 EPP transferTrade request command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. The `request` transferTrade command is used to change the holder of a domain[4]. The operation is equivalent to the domain create command, but requires the domain to exist in another registrar's view and be linked to holder representing another entity than the specified new one. Until the transferTrade is executed by the scheduler the transferTrade request may be cancelled using `cancel` transferTrade command (described in section 5.6.17).

#### Client request

The client issues a domain transfer-trade request to change holder of a domain and obtain sponsorship for the domain.

**Note**: By default a domain will be transferred as active and needs two nameservers. To reserve a domain name, the inactive status flag must be added in the extension part.

```
<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">
```

[4]EPP specifies that it is possible to update the holder reference of a domain object with the domain update command, but a specific transferTrade command makes it possible that other attributes of a domain object can still be updated while the domain is in transferTrade (the transferTrade procedure can last several days) and change the sponsoring registrar. This is made possible by the fact that the domain object is not switched to some pendingTransferTrade state while it is in transferTrade.

```
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:transferTrade op="request">
               <dnslu:domain>
                  <dnslu:name>xn--caf-dma.lu</dnslu:name>
                  <dnslu:ns>
                     <dnslu:hostObj>ns1.restena.lu</dnslu:hostObj>
                     <dnslu:hostObj>ns2.restena.lu</dnslu:hostObj>
                     <dnslu:hostObj>ns3.restena.lu</dnslu:hostObj>
                  </dnslu:ns>
                  <dnslu:registrant>H100</dnslu:registrant>
                  <dnslu:contact type="admin">C100</dnslu:contact>
                  <dnslu:contact type="tech">C100</dnslu:contact>
                  <dnslu:status s="clientHold"/>
                  <dnslu:idn>café.lu</dnslu:idn>
                  <dnslu:trDate>2004-06-30T12:00:00Z</dnslu:trDate>
               </dnslu:domain>
            </dnslu:transferTrade>
            <dnslu:clTRID>ABC-12345</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain transfer-trade request

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **hostObj** | (unbounded) list of nameserver for the domain |
| **registrant** | ID of domain holder |
| **contact[@type=admin]** | ID of Administrative contact |
| **contact[@type=tech]** | ID of Technical contact |
| **status** | (extension, unbounded) status flags (DNS-LU and EPP status flags) |
| **idn** | (extension) Must be IDN equivalent of `xn--` name, UTF-8<br>Not allowed for non-IDN domain names |
| **trDate** | (optional) Explicit date at which trade should be executed |
| **clTRID** | Client transaction ID |

### Server response

The server answers with result code 2106 if a domain with the given name is sponsored by the registrar itself. Code 2300 is returned if the domain is already pending trade, transfer, transfer-trade or transfer-restore.

On successful transfer-trade request of the domain a result code of 1001 is returned. A poll message will be issued when transfer-trade is aborted/executed. Current status of the transfer-trade can be checked with transfer query command described later on. The response contains a trnTraData element that gives information on the requested transfer-trade operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0␣epp-1.0.xsd">
   <response>
      <result code="1001">
         <msg>Command completed successfully; action pending</msg>
      </result>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:trnTraData>
```

```
                  <dnslu:domain>
                     <dnslu:name>xn--caf-dma.lu</dnslu:name>
                     <dnslu:trStatus>pending</dnslu:trStatus>
                     <dnslu:reID>restena-id</dnslu:reDate>
                     <dnslu:reDate>2004-09-08T11:39:41Z</dnslu:reDate>
                     <dnslu:acDate>2004-09-15T11:39:41Z</dnslu:reDate>
                     <dnslu:trDate>2004-09-18T10:00:00Z</dnslu:trDate>
                  </dnslu:domain>
               </dnslu:trnTraData>
            </dnslu:resData>
         </dnslu:ext>
      </extension>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>30C7B5B-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain transfer-trade request response

All fields are printable US-ASCII

| | |
|---|---|
| **name** | Domain name |
| **trStatus** | Transfer-trade status (see EPP transfer status for possible values) |
| **reID** | ID of registrar asking for transfer-trade |
| **reDate** | Date when transfer-trade was requested |
| **acDate** | Date when administrative contact approved/rejected transfer-trade |
| **trDate** | Date when transfer-trade should happen (is possible) |
| **clTRID** | Client transaction ID |
| **svTRID** | Server transaction ID |

### 5.6.16 EPP transferTrade query command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. It provides a query operation that allows an EPP client to determine real-time status of ongoing transferTrade requests (that have been requested using the EPP domain transferTrade request command described in section 5.6.15).

#### Client request

The client issues a domain transfer-trade query request check transfer-trade status for a domain.

```
<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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:transferTrade op="query">
               <dnslu:domain>
                  <dnslu:name>domain.lu</dnslu:name>
               </dnslu:domain>
            </dnslu:transferTrade>
            <dnslu:clTRID>ABC-12345</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain transfer-trade query request

**Server response**

The server answers with result code 2303 if no domain with given name exists, 2301 if domain is not being transfer-traded.

On successful querrying of the transfer-trade status a result code of 1000 is returned and transfer-trade information is returned as in transfer-trade request response.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:trnTraData>
                  <dnslu:domain>
                     <dnslu:name>domain.lu</dnslu:name>
                     <dnslu:trStatus>pending</dnslu:trStatus>
                     <dnslu:reID>restena-id</dnslu:reDate>
                     <dnslu:reDate>2004-09-08T11:39:41Z</dnslu:reDate>
                     <dnslu:acDate>2004-09-15T11:39:41Z</dnslu:reDate>
                     <dnslu:trDate>2004-09-18T10:00:00Z</dnslu:trDate>
                  </dnslu:domain>
               </dnslu:trnTraData>
            </dnslu:resData>
         </dnslu:ext>
      </extension>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>30C7B5B-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain transfer-trade query response

See transfer-trade request response for details.

### 5.6.17 EPP transferTrade cancel command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. It provides a cancel operation that allows an EPP client to cancel a domain transferTrade (that has been requested using the EPP domain transferTrade request command described in section 5.6.15).

**Client request**

The client issues a domain transfer-trade cancel request cancel a domain transfer-trade (**Note**: cancelled/aborted transfer-trades get charged).

```
<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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:transferTrade op="cancel">
               <dnslu:domain>
                  <dnslu:name>domain.lu</dnslu:name>
               </dnslu:domain>
            </dnslu:transferTrade>
            <dnslu:clTRID>ABC-12345</dnslu:clTRID>
```

```
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain transfer-trade cancel request

#### Server response

The server answers with result code 2303 if no domain with given name exists, 2301 if cancelling transfer-trade is not possible because domain is not being transfer-traded.

On successful cancelling of the transfer-trade a result code of 1000 is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>9478978-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain transfer-trade cancel response

### 5.6.18 EPP transferRestore request command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. The `request` transferRestore command is used to change registrar while restoring a pendingDelete domain. The operation is equivalent to the domain transfer command, but requires the domain to be pendingDelete. Until the transferRestore is executed by the scheduler the transferRestore request may be cancelled using `cancel` transferRestore command (described in section 5.6.20).

#### Client request

The client issues a domain transfer restore request to obtain sponsorship for a pendingDelete domain.

For each contact, when permitted by server configuration a copy of current data can be requested with a new contact ID prefixed by symbol 'ª' (Unicode 00AA, "FEMININE ORDINAL INDICATOR"). A contact ID 'CTID' requests use of existing contact with ID 'CTID'; a contact ID 'ªCTID' requests use of non-existing contact with ID 'CTID'. Please read section 3.2.21 about copy of contacts for usage restrictions.

**Note**: By default a domain will be transferred as active and needs two nameservers. To reserve a domain name, the inactive status flag must be added in the extension part.

```
<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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:transferRestore op="request">
               <dnslu:domain>
                  <dnslu:name>xn--caf-dma.lu</dnslu:name>
```

```
                  <dnslu:ns>
                     <dnslu:hostObj>ns1.restena.lu</dnslu:hostObj>
                     <dnslu:hostObj>ns2.restena.lu</dnslu:hostObj>
                     <dnslu:hostObj>ns3.restena.lu</dnslu:hostObj>
                  </dnslu:ns>
                  <dnslu:registrant>H100</dnslu:registrant>
                  <dnslu:contact type="admin">C100</dnslu:contact>
                  <dnslu:contact type="tech">C100</dnslu:contact>
                  <dnslu:status s="clientHold"/>
                  <dnslu:idn>café.lu</dnslu:idn>
                  <dnslu:trDate>2004-06-30T12:00:00Z</dnslu:trDate>
               </dnslu:domain>
            </dnslu:transferRestore>
            <dnslu:clTRID>ABC-12345</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain transfer restore request

All fields are printable US-ASCII, except IDN name

| | |
|---|---|
| **name** | Domain name |
| **hostObj** | (unbounded) list of nameserver for the domain |
| **registrant** | ID of domain holder |
| **contact[@type=admin]** | ID of Administrative contact |
| **contact[@type=tech]** | ID of Technical contact |
| **status** | (extension, unbounded) status flags (DNS-LU and EPP status flags) |
| **idn** | (extension) Must be IDN equivalent of `xn--` name, UTF-8<br>Not allowed for non-IDN domain names |
| **trDate** | (optional) Explicit date at which trade should be executed |
| **clTRID** | Client transaction ID |

### Server response

The server answers with result code 2106 if a domain with the given name is sponsored by the registrar itself. Code 2300 is returned if the domain is already pending trade, transfer, transfer-trade or transfer-restore.

On successful transfer-restore request of the domain a result code of 1001 is returned. A poll message will be issued when transfer-restore is aborted/executed. Current status of the transfer-restore can be checked with transfer-restore query command described later on. The response contains a trnResData element that gives information on the requested transfer-restore operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0␣epp-1.0.xsd">
   <response>
      <result code="1001">
         <msg>Command completed successfully; action pending</msg>
      </result>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:trnResData>
                  <dnslu:domain>
                     <dnslu:name>xn--caf-dma.lu</dnslu:name>
                     <dnslu:trStatus>pending</dnslu:trStatus>
                     <dnslu:reID>restena-id</dnslu:reDate>
                     <dnslu:reDate>2004-09-08T11:39:41Z</dnslu:reDate>
                     <dnslu:acDate>2004-09-15T11:39:41Z</dnslu:reDate>
                     <dnslu:trDate>2005-09-08T11:39:41Z</dnslu:reDate>
```

```
                <dnslu:trDate>2004-09-18T10:00:00Z</dnslu:trDate>
             </dnslu:domain>
          </dnslu:trnResData>
       </dnslu:resData>
    </dnslu:ext>
    </extension>
    <trID>
       <clTRID>ABC-12345</clTRID>
       <svTRID>30C7B5B-DNSLU</svTRID>
    </trID>
  </response>
</epp>
```

Sample domain transfer-restore request response

All fields are printable US-ASCII

| | |
|---|---|
| **name** | Domain name |
| **trStatus** | Transfer-restore status (see EPP transfer status for possible values) |
| **reID** | ID of registrar asking for transfer-restore |
| **reDate** | Date when transfer-restore was requested |
| **acDate** | Date when administrative contact approved/rejected transfer-restore |
| **trDate** | Date when transfer-restore should happen (is possible) |
| **clTRID** | Client transaction ID |
| **svTRID** | Server transaction ID |

### 5.6.19 EPP transferRestore query command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. It provides a query operation that allows an EPP client to determine the real-time status of ongoing transferRestore requests (that have been requested using the EPP domain transferRestore request command described in section 5.6.18).

#### Client request

The client issues a domain transfer-restore query request check transfer-restore status for a domain.

```
<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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:transferRestore op="query">
               <dnslu:domain>
                  <dnslu:name>domain.lu</dnslu:name>
               </dnslu:domain>
            </dnslu:transferRestore>
            <dnslu:clTRID>ABC-12345</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain transfer-restore query request

#### Server response

The server answers with result code 2303 if no domain with given name exists, 2301 if domain is not being transfer-restored.

On successful querrying of the transfer-restore status a result code of 1000 is returned and transfer-restore information is returned as in transfer-restore request response.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>
      <extension>
         <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                    xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
            <dnslu:resData>
               <dnslu:trnResData>
                  <dnslu:domain>
                     <dnslu:name>domain.lu</dnslu:name>
                     <dnslu:trStatus>pending</dnslu:trStatus>
                     <dnslu:reID>restena-id</dnslu:reDate>
                     <dnslu:reDate>2004-09-08T11:39:41Z</dnslu:reDate>
                     <dnslu:acDate>2004-09-15T11:39:41Z</dnslu:reDate>
                     <dnslu:trDate>2004-09-18T10:00:00Z</dnslu:trDate>
                  </dnslu:domain>
               </dnslu:trnResData>
            </dnslu:resData>
         </dnslu:ext>
      </extension>
      <trID>
         <clTRID>ABC-12345</clTRID>
         <svTRID>30C7B5B-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain transfer-restore query response

See transfer-restore request response for details.

### 5.6.20 EPP transferRestore cancel command

This command is an extension to standard EPP - Domain name mapping, rfc3731[3]. It provides a cancel operation that allows an EPP client to cancel a domain transferRestore (that has been requested using the EPP domain transferRestore request command described in section 5.6.18).

#### Client request

The client issues a domain transfer-restore cancel request cancel a domain transfer-restore (**Note**: cancelled/aborted transfer-retores get charged).

```
<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">
   <extension>
      <dnslu:ext xmlns:dnslu="http://www.dns.lu/xml/epp/dnslu-1.0"
                 xsi:schemaLocation="http://www.dns.lu/xml/epp/dnslu-1.0␣dnslu-1.0.xsd">
         <dnslu:command>
            <dnslu:transferRestore op="cancel">
               <dnslu:domain>
                  <dnslu:name>domain.lu</dnslu:name>
               </dnslu:domain>
            </dnslu:transferRestore>
            <dnslu:clTRID>ABC-12345</dnslu:clTRID>
         </dnslu:command>
      </dnslu:ext>
   </extension>
</epp>
```

Sample domain transfer-restore cancel request

**Server response**

The server answers with result code 2303 if no domain with given name exists, 2301 if cancelling transfer-restore is not possible because domain is not being transfer-restored.

On successful cancelling of the transfer-restore a result code of 1000 is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2000/10/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>9478978-DNSLU</svTRID>
      </trID>
   </response>
</epp>
```

Sample domain transfer-restore cancel response

## 5.7 EPP result codes

DNS-LU does not add additional result codes. The result codes 2300 (Object pending transfer) and 2301 (Object not pending transfer) have been overloaded. The term 'transfer' has to be seen as transfer, trade, transfer-trade or transfer-restore depending on context. (See also human readable reasons in result elements).

## 5.8 Troubleshooting

### 5.8.1 Transport issues

In order to determine the exact cause of transport issues it is recommended to analyse the communication at network layer. Some sniffer or raw dumping of socket data should be used for this. Note that especially with RFC3734 transport there are
0x00 bytes in the TCP stream that must be handled appripriately.

### 5.8.2 Charset issues

When using DNS-LU raw XML stream transport you need to make sure that all text (XML souce) is sent UTF-8 encoded. In addition the XML header (if present) must indicate UTF-8 as encoding.

When using RFC3734 transport you may use the encoding of your choice but must indicate that encoding in the XML header.

### 5.8.3 XML validations issues

Error responses for XML validation issues include all required information needed to identify the XML error. Please check your XML message and the relevant schema in order to correct your request. If the error is about trailing data or incorrect first character you are probably facing transport issues. Charset handling errors are in most cases also reported as Syntax errors, for these, please check above.

### 5.8.4 Command Failed

When you get a command failed response from EPP there are multiple possible reasons. When a human readable reason is provided it should give an indication on what is wrong.

A reason like "Failed to parse EPP message of type 'EPPDomainUpdate'" indicates that the command (domain update in this case) passed the XML validation but was incomplete.

In other cases the command should be tried again some time later. If the error persists, please contact technical support.

# Chapter 6

# Registrar interface specification

The registrar interface provides a read-only view to the part of registry database that concerns the registrar. This view includes credit status, transaction list and listings of the objects in addition to the EPP info equivalent.

This chapter is work in progress, check a later version of this document for complete information.

# Part IV

# Appendices

# Appendix A

# Divergences with EPP

## A.1 Object policy

The concept of domain, host and contact objects is defined in EPP. We list a few examples of divergences concerning the three object types below.

- As for the DNS-LU policy, EPP specifies that every object belongs to one registrar. But unlike the DNS-LU policy, EPP specifies that every registrar can update any object, whether he is the owner of the object or not.
- While in the DNS-LU policy only the domain object identifiers, i.e. the domain object names, have to be registry-unique, EPP specifies that every object identifier has to be registry-unique. This results from the fact that in EPP every object can be used by any registrar.
- Unlike the DNS-LU policy, EPP does not distinguish between contact objects that refer to a holder and contact objects that refer to a contact, even if, as in the DNS-LU policy, a contact object can be used as a holder, an administrative or technical contact. In EPP, the attributes of a holder and the attributes of a contact are the same.
- In the DNS-LU policy, there is no authorization information for contact objects.

# Appendix B

# Errata

A list of corrections and changes over the revisions of this document can found below.

o **Revision 2.0.7**

- EPP Server: Added contact copy on transfer and transfer-restore
- EPP Server: fix date fields in examples to include the required time part
- EPP Server: improve explanation of disclosure flags

o **Revision 2.0.6**

- EPP Server: Added RFC3734 support and SSL/TLS support
- EPP Start TLS: Documentation for new command.
- Added missing poll message type 26
- EPP Domain Create: Added creData in response sample with matching documentation.
- EPP Server: added some troubleshooting suggestions
- Registrar Interface: corrected mention of webforms for EPP operations

o **Revision 2.0.5**

- EPP Domain Transfer: Fixed incorrect duplicate domain name in request.

o **Revision 2.0.4**

- EPP Domain Info: Fixed missing elements in response-extension.

o **Revision 2.0.3**

- EPP Poll Request: Added missing list of defined poll message types.

o **Revision 2.0.2**

- EPP Command ContactUpdate: Fixed incorrect description for update of contacts (Holder in field list, missing voice and fax, wrong description for email).

o **Revision 2.0.1**

- EPP Command HostDelete: Fixed typo in sample code where host:delete was embedded in EPP info command instead of EPP delete command.

- EPP Command DomainTransfer: Fixed error in sample code where transfer-cancel was shown as DNS-LU extension though it's standard EPP.

# Bibliography


[1] S. Hollenbeck. *Generic Registry-Registrar Protocol Requirements*. RFC 3375, September 2002.

[2] S. Hollenbeck. *Extensible Provisioning Protocol (EPP)*. RFC 3730, March 2004.

[3] S. Hollenbeck. *Extensible Provisioning Protocol (EPP) Domain Name Mapping*. RFC 3731, March 2004.

[4] S. Hollenbeck. *Extensible Provisioning Protocol (EPP) Host Mapping*. RFC 3732, March 2004.

[5] S. Hollenbeck. *Extensible Provisioning Protocol (EPP) Contact Mapping*. RFC 3733, March 2004.

[6] S. Hollenbeck. *Extensible Provisioning Protocol (EPP) Transport Over TCP*. RFC 3734, March 2004.

[7] S. Hollenbeck. *Guidelines for Extending the Extensible Provisioning Protocol (EPP)*. RFC 3735, March 2004.