SWITCH, Werdstrasse 2, P.O. Box, CH-8021 Zürich
www.nic.ch

EPP Manual

Version 2.3.3
23 January 2023

SWITCH

SWITCH, Werdstrasse 2, P.O. Box, CH-8021 Zürich
www.nic.ch

Contents

SWITCH, Werdstrasse 2, P.O. Box, CH-8021 Zürich
www.nic.ch

1 Management Summary

The EPP Protocol (Extensible Provisioning Protocol) is an international standard for the administration of domain names. It allows the information that has to be exchanged between Registrars and the registry operator to be sent via a secure and permanent connection. EPP allows Registrars to integrate the administration of domain names in their existing system.

This document contains all the information that is required for connecting up the Registrar's EPP client to the EPP Server at the Registry SWITCH. EPP has been implemented extensively according to the international standard; the SWITCH-specific deviations are specially marked. 

2 Introduction

SWITCH is the office responsible for the registration and administration of the country-code-Top-Level Domain Names (ccTLDs) for Switzerland (.ch) and Liechtenstein (.li). One of the main tasks this involves is the registration of second-level domain names under these ccTLDs, which will simply be referred to as “domain names” in what follows.

2.1 EPP standard and legal fundaments

This Manual is based on the following documents. Detailed examples of EPP commands are given in the RFCs listed below:

Reference	Document
[01]	RFC 5730: Extensible Provisioning Protocol (EPP) (http://www.ietf.org/rfc/rfc5730.html)
[02]	RFC 5731: Extensible Provisioning Protocol (EPP) Domain Name Mapping (http://www.ietf.org/rfc/rfc5731.html)
[03]	RFC 5732: Extensible Provisioning Protocol (EPP) Host Mapping (http://www.ietf.org/rfc/rfc5732.html)
[04]	RFC 5733: Extensible Provisioning Protocol (EPP) Contact Mapping (http://www.ietf.org/rfc/rfc5733.html)
[05]	RFC 5734: Extensible Provisioning Protocol (EPP) Transport Over TCP (http://www.ietf.org/rfc/rfc5734.html)
[06]	SWITCH Registrar Agreement, including Annexes
[07]	General Terms and Conditions (GTCs) for the registration and administration of domain names under “.ch” and “.li” The latest version is always available at www.nic.ch.
[08]	RFC 5910: Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP) (http://www.ietf.org/rfc/rfc5910.html)
[09]	RFC 3915: EPP Grace Period Mapping - Redemption Grace Period (RGP) (http://www.ietf.org/rfc/rfc3915.html)
[10]	RFC 8590: Change Poll Extension for EPP (https://tools.ietf.org/html/rfc8590.html)
[11]	RFC 9038 Extensible Provisioning Protocol (EPP) Unhandled Namespaces (https://www.ietf.org/rfc/rfc9038.html)

- *This symbol denotes important SWITCH-specific technical deviations from the international EPP standard.*

The Registrar Agreement and its Annexes always take precedence over this User Manual.

2.2 Conditions of use

The following conditions must be fulfilled for regular operation of the EPP interface:

- A signed and valid Registrar Agreement
- An IP address for your EPP client that has been activated by SWITCH
- A successfully completed test run (see Paragraph 5.3)

3 Using the EPP interface

An EPP client with a TLS connection with an up to date version of TLS is essential for using the EPP interface. **Valid are only versions from TLS 1.2**. Before a TLS connection can be established, the IP address of the EPP client must already have been activated by SWITCH. Once the TLS connection has been established, only the EPP commands of `login` and `hello` are permitted to begin with. All other EPP commands are only possible after a successful login. SWITCH does not offer a standard EPP client or an EPP library. Any EPP client can be used that satisfies the referenced standards as per Paragraph 2.1.

In addition to the standard RFC, an EPP client must be configured in accordance with the provisions of this Manual and the Registrar Agreement (including its Annexes).

The EPP server supports the UTF-8 unicode character set (see Paragraph 3.2). Communication between the EPP client and the EPP server takes place by means of commands in XML format. The server will immediately send back the corresponding reply. Where there are several commands in succession (pipelining) these will be processed individually, waiting until the reply has been sent to the previous command in each case.

3.1 Options offered by the EPP interface

The EPP interface has two main categories: objects and commands (actions). The objects are:

- domain
- contact
- host

The commands are divided up into protocol commands and object-specific commands.

The following table gives an overview of the protocol commands:

Command	Description
hello	Making contact
login	Logging onto the EPP server
logout	Logging off from the EPP server
poll	Retrieving messages from the poll queue on the EPP server – this is the route by which the EPP user receives notifications from SWITCH.

The following Table gives an overview of the object-specific commands:

Object-specific commands		domain	contact	Host
check	Checks the existence of an object	✓	✓	✓
info	Supplies information on an object	✓	✓	✓
create	Creates a new object	✓	✓	✓
delete	Deletes an object	✓	✓	✓
renew	Renews an object	×	-	-
transfer	Transfers an object	✓	×	(implicit)
update	Updates the data for an object	✓	✓	✓

✓ implemented
× not implemented
(implicit) this command is not provided for in EPP.

3.2 SWITCH-specific general points

Under the terms of Reference (01), all EPP commands are “atomic” and are either processed in full or rejected in full. If an EPP command is submitted twice, the second command will also be processed once again. If a domain name was already registered the first time the command was submitted, an error message will be issued the second time the same domain name is submitted.

When requesting information on an object (contact, domain and host), the data authorised for publication by the Ordinance of 5 November 2014 on Internet Domains (OID, Art. 46) will be returned, unless the EPP user is authorised to see further information under the terms of the Registrar Agreement, e.g. the expiry date of a domain name for the managing Registrar.

In general, any information that is not mandatory and is not used, which the EPP user submits, will be ignored.

- *A status allocated to objects by the EPP client is not supported, such as* `<clientDeleteProhibited>`, `<clientTransferProhibited>` *etc.*
- *Each Registrar is allowed to connect to the EPP server with maximally three sessions simultaneously.*
- *In general, the following UTF-8 characters are supported:*

  - *Basic Latin* *U+0020 - U+007E*
  - *Latin-1 Supplement* *U+00AE - U+00FF U+00A1 - U+00AC*
  - *Latin Extended-A* *U+0100 - U+017F*
  - *€ euro sign* *U+20AC*

- *Dates are given in local time, to the nearest second, stating the difference compared with UTC. For the example given for summer time below, UTC is thus 2007-09-18Z12:32:00.*

Format:	yyyy-mm-ddThh:mi:ss+hh:mi (T instead of Z)
Summer time:	2007-09-18T14:32:00+02:00
Winter time:	2007-12-18T13:32:00+01:00

3.3 Session ended

An EPP session can be ended by the server for the following reasons:

- The EPP session has been inactive for more than 3 hours (“session-timeout”)
- The maximum number of unsuccessful EPP commands has been reached
- Scheduled or non-scheduled maintenance work on the EPP server

4 EPP commands

Each Registrar has a maximum number of EPP commands per minute available to them.The quota is determined by the number of managed domain names. Depending on the load on the registration system, SWITCH may dynamically modify this rate. Once the maximum number of commands per minute has been achieved, the server will delay accepting any further EPP requests.

Number of domain names	Commands per minute
< 1'000	50
up to 100'000	100
Up to 200'000	150
Up to 300'000	200
Up to 400'000	250
Up to 500'000	300
Up to 600'000	350

4.1 Protocol commands

In addition to what is set out below, the protocol commands are described in detail in Reference (1).

4.1.1 login

An EPP session is set up by means of

a) a TLS connection to the EPP server and
b) a subsequent successful login with the EPP `login` command. The username is assigned by SWITCH according to 3.2, except a blank/space character.

The EPP server acknowledges the successful establishment of the TLS connection with a greeting. After the connection has been set up, the client still has to sign on with the `login` command. The EPP server then sends an EPP response. Code 1000 means that the EPP session has been successfully established.

After a successful `login` the EPP response will say how many messages are waiting in the poll queue and what the first message is about. Until the EPP client has sent a `login`, only `hello` is supported.

The EPP login password can also be altered at the same time as the `login` command is submitted:

```
<pw>old-password</pw>
 <newPW>new-password</newPW>
```


The EPP login password must be 10 - 16 characters in length. In addition a password must contain lowercase und uppercase letters, at least one digit and one special character[1]. Particular attention should be paid to the provisions of the GTCs when it comes to the careful selection of passwords and to keeping passwords safe and confidential.

Example of a login command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
 <command>
  <login>
   <clID>YOUR-CLID</clID>
   <pw>ABCDEF</pw>
   <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:contact-1.0</objURI>
    <objURI>urn:ietf:params:xml:ns:host-1.0</objURI>
   </svcs>
  </login>
  <clTRID>ABC.1</clTRID>
 </command>
</epp>
```

Supported Extensions

```
<svcExtension>
  <extURI>urn:ietf:params:xml:ns:secDNS-1.1</extURI>
  <extURI>urn:ietf:params:xml:ns:rgp-1.0</extURI>
  <extURI>urn:ietf:params:xml:ns:changePoll-1.0</extURI>
  <extURI>https://www.nic.ch/epp/balance-1.0</extURI>
</svcExtension>
```

Example of a login response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <msgQ count="1" id="1139047">
      <qDate>2007-09-26T00:00:00+02:00</qDate>
      <msg>Domain transfer completed successfully</msg>
    </msgQ>
    <trID>
      <clTRID>ABC.1</clTRID>
      <svTRID>20071008.13747.27297</svTRID>
    </trID>
  </response>
</epp>
```

TEST-REGISTRAR-A has an unacknowledged message in the poll queue.

- *The EPP server **epp.nic.ch** is accessed with a TLS connection via Port 700.*

[1] Allowed special characters: + - % ( ) = . _

- *The EPP server only allows connections from clients whose IP addresses are registered at SWITCH.*
- *Only secDNS-1.1 is supported*
- *Registrars can now use the DNSSEC extension without SWITCH activating it. By using the DNSSEC extension when logging in, the registrar committes to support DNSSEC via EPP. Incoming transfers of domain names with DNSSEC are only allowed if the DNSSEC extension was configured at login.*

4.1.2 logout

The EPP client must end its session with the `logout` command.

Example of a logout command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
 <epp:command xmlns:epp="urn:ietf:params:xml:ns:epp-1.0">
  <epp:logout />
  <epp:clTRID>ABC.27</epp:clTRID>
 </epp:command>
</epp>
```

Example of a logout response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1500">
      <msg lang="en">Command completed successfully; ending session</msg>
    </result>
    <trID>
      <clTRID>ABC.27</clTRID>
      <svTRID>20071008.13747.27299</svTRID>
    </trID>
  </response>
</epp>
```

4.1.3 hello

The EPP client can use the `hello` command to see whether the EPP session is still active. The EPP server answers a `hello` command with the greeting.

Example of a hello command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <hello/>
</epp>
```

Example of a hello response (greeting)

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <greeting>
    <svID>SWITCH_EPP_Server</svID>
    <svDate>2007-10-09T08:44:14+02:00</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:contact-1.0</objURI>
      <objURI>urn:ietf:params:xml:ns:host-1.0</objURI>
      <svcExtension>
        <extURI>urn:ietf:params:xml:ns:secDNS-1.1</extURI>
        <extURI>urn:ietf:params:xml:ns:rgp-1.0</extURI>
        <extURI>urn:ietf:params:xml:ns:changePoll-1.0</extURI>
        <extURI>https://www.nic.ch/epp/balance-1.0</extURI>
        <extURI>urn:ietf:params:xml:ns:epp:unhandled-namespaces-1.0</extURI>
    </svcExtension>
      </svcMenu>
      <dcp>
        <access>
          <personalAndOther />
        </access>
        <statement>
          <purpose>
            <admin />
            <other />
            <prov />
          </purpose>
          <recipient>
            <ours />
            <public />
          </recipient>
          <retention>
            <legal />
          </retention>
        </statement>
      </dcp>
    </greeting>
</epp>
```

4.1.4 poll

The messages in the poll queue must be retrieved one after the other and acknowledged. As soon as the first message has been acknowledged, the second one can be retrieved, etc. Messages that have been acknowledged are deleted from the poll queue.

→ Not confirmed messages are deleted from the poll queue after 30 days and can no longer be retrieved. Therefore, it is important that the registrar retrieves the messages the poll queue regularly in order not to miss important information.

Example of a poll command for retrieving a message

```
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <poll op="req"/>
    <clTRID>Registrar_00_2</clTRID>
  </command>
</epp>
```

Example of a poll command for confirming a message

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <poll op="ack" msgID="1139047"/>
    <clTRID>Registrar_00_3</clTRID>
  </command>
</epp>
```

Example of a poll response for confirming a message

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <msgQ count="0" id="1139047"/>
    <trID>
      <clTRID>Registrar_00_3</clTRID>
      <svTRID>20071008.13689.27043</svTRID>
    </trID>
  </response>
</epp>
```

4.1.4.1 Domain Transfer Completed message

This message is put in the poll queue of the Registrar to date once a domain transfer has been completed.

Attribute	Comment
domain:name	Domain name
domain:trStatus	“serverApproved”
domain:reID	contact:id of the requester
domain:reDate	Date of request
domain:acID	“ZERO”
domain:acDate	Execution date
domain:exDate	This is omitted.

Example of a poll response after successful transfer

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1301">
      <msg lang="en">Command completed successfully; ack to dequeue</msg>
    </result>
    <msgQ count="1" id="1139047">
      <qDate>2007-09-26T00:00:00+02:00</qDate>
      <msg> Domain transfer completed successfully</msg>
    </msgQ>
    <epp:resData xmlns:epp="urn:ietf:params:xml:ns:epp-1.0">
      <domain:trnData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
             <domain:name>yourname.ch</domain:name>
             <domain:trStatus>serverApproved</domain:trStatus>
             <domain:reID>RegistrarB</domain:reID>
             <domain:reDate>2007-09-18T22:43:00+02:00</domain:reDate>
             <domain:acID>NULL</domain:acID>
             <domain:acDate>2007-09-18T22:43:00+02:00</domain:acDate>
          </domain:trnData>
        </epp:resData>
    <trID>
      <clTRID>Registrar_00_2</clTRID>
      <svTRID>20071008.13688.27039</svTRID>
    </trID>
  </response>
</epp>
```

4.1.4.2 Domain Delete message

This message is put into the poll queue of the registrar after a manual deletion of the domain name by the registry. Such deletion can occur in special circumstances like domain name abuse.

Attribute	Comment
domain:name	Domain Name
domain:upDate	Execution Date
domain:clID	deletedBy-nic.ch (indicating Domain Delete Message until Change-Poll is implemented)

Example of a poll response after deletion by the registry

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1301">
      <msg lang="en">Command completed successfully; ack to dequeue</msg>
    </result>
    <msgQ count="1" id="46535949">
      <qDate>2019-01-28T16:14:47+01:00</qDate>
    </msgQ>
    <resData>
      <domain:infData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>delete-me11.ch</domain:name>
        <domain:roid>D6006352-SWITCH</domain:roid>
        <domain:status s="serverHold" lang="en" />
        <domain:status s="serverRenewProhibited" lang="en" />
        <domain:status s="serverTransferProhibited" lang="en" />
        <domain:status s="inactive" lang="en" />
        <domain:registrant>CH-MYTECH</domain:registrant>
        <domain:clID>Test-Registrar-X</domain:clID>
        <domain:upDate>2019-01-28T16:14:47+01:00</domain:upDate>
      </domain:infData>
    </resData>
    <extension>
      <changePoll:changeData xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
        <changePoll:operation>delete</changePoll:operation>
        <changePoll:date>2019-01-28T16:14:47+01:00</changePoll:date>
        <changePoll:svTRID>20190128.34733373</changePoll:svTRID>
        <changePoll:who>SWITCH manual delete</changePoll:who>
        <changePoll:reason>domain name abuse</changePoll:reason>
      </changePoll:changeData>
    </extension>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20190128.75290441.758467418</svTRID>
    </trID>
  </response>
</epp>
```

4.1.4.3 “Automated DNSSEC Provisioning” message

This message is put into the EPP poll queue of the registrar after a change on the DS records of its domain name was done over the „Automated DNSSEC Provisioning“ process.

You will find more information about the different variations of rendering this message and how to trigger examples of them in our test environment in the „Guidelines for CDS processing at SWITCH“:

https://www.nic.ch/export/shared/.content/files/SWITCH_CDS_Manual_en.pdf

and on the “CDS Status Check” website:

https://www.nic.ch/de/faqs/dnssec/cds/

Attribute	Comment
changePoll:reason	Indicates bootstrap, rollover or delete of DS records
changePoll:who	SWITCH CDS: indicates Automated DNSSEC provisioning process change
changePoll:date	Execution Date

Example of a Poll Message after change by “Automated DNSSEC provisioning” process

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
 <response>
  <result code="1301">
   <msg lang="en">Command completed successfully; ack to dequeue</msg>
  </result>
  <msgQ count="1" id="120030094">
   <qDate>2018-06-19T00:00:00+02:00</qDate>
  </msgQ>
  <epp:resData xmlns:epp="urn:ietf:params:xml:ns:epp-1.0">
   <domain:infData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
    <domain:name>rollover-ea48.ch</domain:name>
    <domain:roid>D5920069-SWITCH</domain:roid>
    <domain:status s="ok" />
    <domain:registrant>123536</domain:registrant>
    <domain:clID>cds message</domain:clID>
    <domain:upDate>2018-06-21T11:49:54+02:00</domain:upDate>
   </domain:infData>
  </epp:resData>
  <extension>
   <changePoll:changeData xmlns:changePoll="urn:ietf:params:xml:ns:changePoll-1.0">
    <changePoll:operation>update</changePoll:operation>
    <changePoll:date>2018-06-21T11:49:54+02:00</changePoll:date>
    <changePoll:svTRID>20180621.34663867</changePoll:svTRID>
    <changePoll:who>SWITCH CDS: see https://www.nic.ch/de/faqs/dnssec/cds/</changePoll:who>
    <changePoll:reason>Rollover of DNSSEC Digest</changePoll:reason>
   </changePoll:changeData>
   <secDNS:infData xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
    <secDNS:dsData>
     <secDNS:keyTag>20011</secDNS:keyTag>
     <secDNS:alg>8</secDNS:alg>
     <secDNS:digestType>2</secDNS:digestType>
     <secDNS:digest>AAAA9AB3E7D203FF7923B8773599E248717F1DC79A9BEF09D8981B13AB7A049E</secDNS:digest>
    </secDNS:dsData>
    <secDNS:dsData>
     <secDNS:keyTag>12123</secDNS:keyTag>
     <secDNS:alg>7</secDNS:alg>
     <secDNS:digestType>4</secDNS:digestType>
<secDNS:digest>AAAA54840FBBB6F4270F8B6D8C06C6A2B3152E55D2E9F81132130E507829B6D24FA56A4E074B4692DDC46F512B0
48AAC</secDNS:digest>
    </secDNS:dsData>
```

```
   </secDNS:infData>
  </extension>
  <trID>
   <clTRID>EA03_RY_10_2_C</clTRID>
   <svTRID>$svTRID$</svTRID>
  </trID>
 </response>
</epp>
```

4.1.5 Change-Poll Extension

For some domain name changes made by SWITCH, we send an EPP Poll message to the registrar for information.

These include the updated domain status (<epp:resData><domain:infData><domain:status>) and the ChangePoll extension.

The Change Poll Extension is an extension of the EPP Poll messages to provide additional information about changes to EPP objects. (what, when, who and why)

The applications concerned are:

- Deletions (manually or through our malware process)
- Activation / deactivation of domain names (manually or through our malware process)
- Changes to the name servers
- Reset of the transfer code
- CDS see chapter 4.1.4.3 Automated DNSSEC Provisioning

The implementation of ChangePoll is based on https://tools.ietf.org/html/rfc8590

→ For an easier automated processing we recommend to login with the Change-Poll Extension to EPP.

Otherwise the corresponding XML element will still be provided in the <extValue> element (for possible subsequent processing).
https://datatracker.ietf.org/doc/draft-ietf-regext-unhandled-namespaces/ (in Progress)

Otherwise, the XML element with the namespace "urn:ietf:params:xml:ns:changePoll-1.0" will be included in the <extValue> according to RFC 9038 Unhandled Namespaces. See also chapter Unhandled Namespaces Extension.

4.1.6 Unhandled Namespace extension

EPP, as defined in RFC 5730, comprises a procedure to determine the objects (domain, host, contact) and service extensions (RGP, DNSSEC, etc.) valid during a session. The objects and extensions are identified with namespace URIs.

If an object or service extension is supported by the server (listed in the server greeting) but not yet supported by the client (not listed in the login command), then its namespace is an "unhandled namespace". RFC 9038 Unhandled Namespaces defines a practice for the server to send data from unhandled namespaces to the client in compliance with RFC 5730.
This informs the client of the existence of such namespaces and allows them to be processed automatically at a later time.

A server that has implemented this practice lists <extURI>urn:ietf:params:xml:ns:epp:unhandled-namespaces-1.0</extURI> in the server greeting.

A client that lists <extURI>urn:ietf:params:xml:ns:epp:unhandled-namespaces-1.0</extURI> in the EPP login signals that it is checking the server responses for unhandled namespaces.

4.1.7 Account balance query

With the command below the current account balance of the registrar can be determined. Since this is an extension, this must be specified during login so that the response can be created regularly. If you do not specify the extension at login, you will get the following response:

```
<result code="2307">
  <msg lang="en">Unimplemented object service</msg>
</result>
```

The schema for command and response is available at the following link:
https://www.nic.ch/epp/balance-1.0

Example Command account balance query

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <info>
      <balance:info
        xmlns:balance="https://www.nic.ch/epp/balance-1.0"/>
    </info>
    <clTRID>ABC</clTRID>
  </command>
</epp>
```


Example Response with the current account balance

```
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <infData xmlns="https://www.nic.ch/epp/balance-1.0">
        <balance>1234.50</balance>
        <currency>CHF</currency>
      </infData>
    </resData>
    <trID>
      <clTRID>ABC</clTRID>
      <svTRID></svTRID>
    </trID>
  </response>
</epp>
```

4.2 Domain commands

4.2.1 General information

In addition to what is set out below, the `domain` commands are described in detail in Reference (1).

- The holder of a domain name is saved in the `domain:registrant` box.
- The domain name transfer code may only be given to the holder of the domain name.

- *Mention is made of a `renew` command in the EPP specification. This is not implemented, since a renew is performed by the server. A `renew` is rejected with error code 2101 ("unimplemented command").*
- *Each domain name can be optionally allocated a maximum of one `type="tech"` contact.*

domain attribute	mandatory	Update possible	Comment
name	Yes	No	
roid	Yes	No	Dnnnnnnnn-SWITCH, supplied by the registry
status	No	No	
registrant	Yes	Yes	contact:id
contact type=tech	No	Yes	contact:id
hostObj	No	Yes	
clID	Yes	No	Set by the registry
crID			This is omitted.
crDate		No	Set by the registry
exDate		No	Set by the registry
upID			This is omitted.
upDate		No	This is omitted.
trDate	No	No	This is omitted.
authinfo	No	Yes	Blank spaces ";" and "," are not permitted.

4.2.2 DNSSEC Extension

DNSSEC can be used on an optional basis and is deployed for signing domain names.

DNSSEC attribut	mandatory	Comment
alg	Yes	algorithm 8, 10, 13, 14, 15, 16
digestType	Yes	2 or 4
digest	Yes	max. 96 characters
flags	No	256 or 257
protocol	No	3
keyTag	Yes	0-65535 value range
pubKey (DNSKEY)	No	max. 4096 bits

- The registrar no longer has to be activated separately for the use of DNSSEC. *When logging in, DNSSEC (urn:ietf:params:xml:ns:secDNS-1.1) should only be declared if the registrar supports DNSSEC via EPP.*
- *SWITCH does not make delegation checks and does not check if signed domain names are accessible.*
- *keyData entries are optional and will be saved in the database if sent.*
- *If keyData is used, the flags, protocol, alg and pubKey attributes are mandatory.*
- *Only secDNS-1.1 DS Data Interface is supported (Reference 8).*
- *Up to 20 DNSSEC entries per domain name*
- *The attributes <secDNS:update urgent="true"> and <secDNS:maxSigLife> according to rfc5910 are not supported. Requests containing these attributes are answered with the ResultCode "2102 Unimplemented option".*
- *Please note that algorithms 5 (RSASHA1),7 (RSASHA1-NSEC3-SHA1) and digest type 1 (SHA1) are no longer supported by SWITCH.*

4.2.3 RGP Extension

RGP can be used on an optional basis to restore deleted domain names.

- *RGP must be declared when logging in: urn:ietf:params:xml:ns:rgp-1.0*
- *If the domain name belongs to the Registrar submitting the query, the command domain:info can be used to enquire whether the domain name is within the redemption period*
- *There is no pendingRestore RGP status. If the restore command has been successful, then the EPP status of the domain name will switch to "OK".*
- *The optional rgp:report element in the rgp:restore element is ignored.*
- *A separate rgp:report command leads to an error, since there is no pendingRestore.*

4.2.4 *domain:*check

Using `domain:check` it is possible to check whether domain names are available. The server will reply with available / not available. Bulk queries are only permitted to a limited extent, as per Reference (06).

- *A maximum of 10 domain names can be queried at a time in an individual command, otherwise the server will answer with error code 2308 (“Data management policy violation”)*
-

Example of a domain check command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <check>
      <domain:check xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>yourname.ch</domain:name>
        <domain:name>yourname.li</domain:name>
        <domain:name>studen-be.ch</domain:name>
        <domain:name>fuerstentum.li</domain:name>
      </domain:check>
    </check>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a domain check response

```
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <domain:chkData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:cd>
          <domain:name avail="0">yourname.ch</domain:name>
          <domain:reason>In use</domain:reason>
        </domain:cd>
        <domain:cd>
          <domain:name avail="1">yourname.li</domain:name>
        </domain:cd>
        <domain:cd>
          <domain:name avail="0">studen-be.ch</domain:name>
          <domain:reason>City-domainname</domain:reason>
        </domain:cd>
        <domain:cd>
          <domain:name avail="0">fuerstentum.li</domain:name>
          <domain:reason>Legal reasons</domain:reason>
        </domain:cd>
      </domain:chkData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20071009.14301.29655</svTRID>
    </trID>
  </response>
</epp>
```

4.2.5 *domain:*info

The `domain:info` command is used to have the data for the domain name disclosed with (see Paragraph 3.2). In addition to the standard data, the `<domain:clID>` box contains the ID of the "sponsoring client", i.e. the current Registrar's ID.

The fields `<domain:registrant>`, `<domain:exDate>` and `<domain:contact>` (if available) will only be shown, if

- the domain name is administered by the requesting registrar
- a valid transfer code is supplied, according to the following example

Example domain info Command with transfer code

```
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <command>
        <info>
            <domain:info
                xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
                <domain:name>fremder-domain.ch</domain:name>
                <domain:authInfo>
                    <domain:pw>authinfo_pwd</domain:pw>
                </domain:authInfo>
            </domain:info>
        </info>
        <clTRID>rq0dr</clTRID>
    </command>
</epp>
```

Example domain info Response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <domain:infData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>fremder-domain.ch</domain:name>
        <domain:roid>D7033341-SWITCH</domain:roid>
        <domain:status s="inactive" lang="en" />
        <domain:registrant>CH-HOLDER-ID</domain:registrant>
        <domain:contact type="tech">CH-TECH-ID</domain:contact>
        <domain:clID>Test-Registrar-X</domain:clID>
        <domain:exDate>2021-11-09T23:59:59+01:00</domain:exDate>
      </domain:infData>
    </resData>
    <trID>
      <clTRID>ABC</clTRID>
      <svTRID>20201110.118733861.1213851368</svTRID>
    </trID>
  </response>
</epp>
```

With a valid transfer code, the response is: `<result code="1000">`

Wrong entries in the optional `<domain:authInfo>` element always lead to the return code:
`<2202 Invalid authorization information>`

A roid attribute provided in `<domain:pw>` is always ignored. The value of `<domain:pw>` must always refer to the queried domain name and must not be empty.

Example of a domain info command self-managed

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <info>
      <domain:info xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>yourname.ch</domain:name>
      </domain:info>
    </info>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a domain info response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <domain:infData xmlns:domain="urn:ietf:params:xml:ns:
       domain-1.0">
        <domain:name>yourname.ch</domain:name>
        <domain:roid>D2586061-SWITCH</domain:roid>
        <domain:status s="ok"/>
        <domain:registrant>5527981</domain:registrant>
        <domain:contact type="tech">5527981</domain:contact>
        <domain:ns>
          <domain:hostObj>ns1.yourname.ch</domain:hostObj>
          <domain:hostObj>ns2.yourname.ch</domain:hostObj>
        </domain:ns>
        <domain:clID>3703709</domain:clID>
        <domain:exDate>2009-01-31T00:00:00+01:00
        </domain:exDate>
      </domain:infData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20080529.27664.913190</svTRID>
    </trID>
  </response>
</epp>
```

For DNSSEC signed domain names

```
<extension>
  <secDNS:infData
   xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
    <dsData xmlns="urn:ietf:params:xml:ns:secDNS-1.1">
      <keyTag>20530</keyTag>
      <alg>13</alg>
      <digestType>2</digestType>
      <digest>CAFFEEBABE00D87A0147EFE9877
       AB7335206ABFF6FA730BD6239D65CBAC7E768</digest>
      <keyData>
        <flags>257</flags>
        <protocol>13</protocol>
        <alg>5</alg>
        <pubKey>L1tqdb8EDyeQ/8RyJL1faHGKBw
        LPTYfZLV8vRN3opa4uY+nyhPghl
        EEbqDLDdCo3ZpywJFlW39vwl03PZJvSzg==</pubKey>
      </keyData>
    </dsData>
  </secDNS:infData>
</extension>
```

Query about a domain name administered by the Registrar who is submitting the query.

Example of a domain info response with "redemptionPeriod" status

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <domain:infData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>switch02.ch</domain:name>
        <domain:roid>D5874403-SWITCH</domain:roid>
        <domain:status s="serverHold" lang="en" />
        <domain:status s="serverRenewProhibited" lang="en" />
        <domain:status s="serverTransferProhibited" lang="en" />
        <domain:registrant>SWITCH01</domain:registrant>
        <domain:ns>
          <domain:hostObj>ns1.switch02.ch</domain:hostObj>
          <domain:hostObj>ns2.switch02.ch</domain:hostObj>
        </domain:ns>
        <domain:host>ns1.switch02.ch</domain:host>
        <domain:host>ns2.switch02.ch</domain:host>
        <domain:clID>Test-X</domain:clID>
        <domain:upDate>2018-01-09T09:06:48+01:00</domain:upDate>
        <domain:exDate>2019-01-08T23:59:59+01:00</domain:exDate>
      </domain:infData>
    </resData>
    <extension>
      <rgp:infData xmlns:rgp="urn:ietf:params:xml:ns:rgp-1.0">
        <rgp:rgpStatus s="redemptionPeriod" lang="en" />
      </rgp:infData>
    </extension>
    <trID>
      <clTRID>DS03_RY_00_2_C</clTRID>
      <svTRID>20180109.75043211.757740352</svTRID>
    </trID>
  </response>
</epp>
```

Query about a domain name administered by the Registrar who is submitting the query.

4.2.6 *domain:*create

`domain:create` is used to register domain names. Prior to this, all the contacts (the holder and the technical contact) must have been entered if they do not already exist. Name servers must also be registered beforehand so that they can be allocated to the domain name.

- `<admin>` *not supported.*
- `<billing>` *not supported (the Registrar is always the billing contact).*

Example of a domain create command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create xmlns:domain="urn:ietf:params:xml:ns:
       domain-1.0">
        <domain:name>yourname.ch</domain:name>
         <domain:ns>
          <domain:hostObj>ns1.yourname.ch</domain:hostObj>
          <domain:hostObj>ns2.yourname.ch</domain:hostObj>
         </domain:ns>
        <domain:registrant>HOLDERCONTACT</domain:registrant>
        <domain:contact
         type="tech">TECHCONTACT</domain:contact>
        <domain:authInfo>
          <domain:pw/>
        </domain:authInfo>
      </domain:create>
    </create>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

DNSSEC Extension

```
<extension>
  <secDNS:create
   xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
    <secDNS:dsData>
      <secDNS:keyTag>20530</secDNS:keyTag>
      <secDNS:alg>13</secDNS:alg>
      <secDNS:digestType>1</secDNS:digestType>
<secDNS:digest>CAFFEEBABE00D87A0147EFE9877
AB7335206ABFF6FA730BD6239D65CBAC7E768</secDNS:digest>
      <secDNS:keyData>
        <secDNS:flags>256</secDNS:flags>
        <secDNS:protocol>3</secDNS:protocol>
        <secDNS:alg>5</secDNS:alg>
        <secDNS:pubKey>Lltqdb8EDyeQ/8RyJL1faHGKBw
         LPTYfZLV8vRN3opa4uY+nyhPghl
       EEbqDLDdCo3ZpywJF1W39vwl03PZJvSzg==</secDNS:pubKey>
     </secDNS:keyData>
    </secDNS:dsData>
  </secDNS:create>
</extension>
```

`keyData` is optional. If sent, the calculation of the `pubKey` has to lead to the `digest`.

Example of a domain create response

```
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <domain:creData
       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>yourname.ch</domain:name>
        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
        <domain:exDate>2001-04-03T22:00:00.0Z</domain:exDate>
      </domain:creData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

4.2.7 *domain:*delete

`domain:delete` is used to delete domain names with immediate effect and put them into redemption. A domain name can be deleted by the authorised Registrar at any time, observing the rules in Reference (06). A deleted domain name can only be re-registered after the transition period as per Reference (07).

A domain name can be deleted even if subordinate name servers (subordinate hosts) are available. These continue to be registered but can no longer be used (see Paragraph 4.4).

Example of a domain delete command

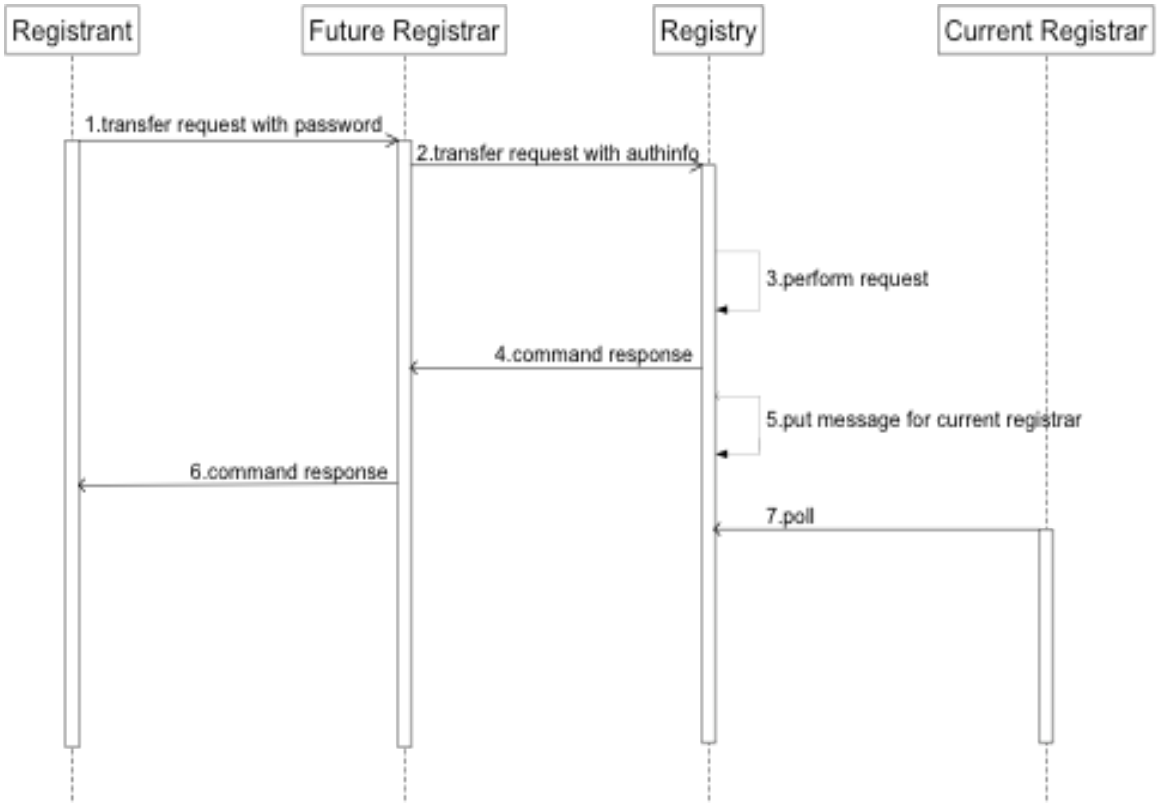
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <delete>
      <domain:delete
       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>yourname.ch</domain:name>
      </domain:delete>
    </delete>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a domain delete response

```
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

4.2.8 *domain:*transfer

The holder can have their domain name administered by a Registrar of their choice. The `domain:transfer` command is used to transfer domain names. If subordinate name servers do exist for the domain name to be transferred, then these will be transferred as well.



The complete domain transfer process in seven steps:

1. transfer request with password

The holder must request a transfer code for their domain name from their current Registrar. As soon as they have obtained the transfer code, they can forward it to the new Registrar.

2. transfer request with authinfo

The new Registrar sends the `domain:transfer` request, with the corresponding transfer code to SWITCH, via EPP.

3. perform request

The EPP server checks the request and transfers the domain name if the transfer code is correct. Any subordinate name servers will similarly be transferred (see Paragraph 4.4.1). After the transfer, the domain name comes under the new Registrar's domain name portfolio. The contact IDs are not supplied with a transfer and need to be replaced with contact IDs created by the new Registrar.

The new Registrar must use `domain:update` to enter a contact ID that they have created for the holder (a technical contact is not mandatory), before further updates are permitted for the domain name. Until this has been done, `check`, `info`, `delete` and `transfer` are the only valid commands.

4. command response

The EPP server sends the new Registrar a confirmation message (EPP response).

5. put message for current registrar

SWITCH puts a new message in the poll queue for the Registrar to date, confirming the transfer of the domain name to a new Registrar or to SWITCH.

6. command response

The new Registrar can inform the holder that the transfer has been completed. SWITCH does not generally make contact with the holder as long as the holder is with a Registrar, except in the cases mentioned in Reference (06).

7. poll

The Registrar to date can retrieve the message generated under Number 5 above from their poll queue.

- *After a successful transfer, the status* `<serverTransferProhibited>` *is assigned. While the domain name has this status, it cannot be transferred to another Registrar. This status lasts 60 days and is then automatically removed.*

- *The Transfer Query command is not required, since the transfer will be carried out immediately if the transfer code is valid. The server will respond to a Transfer Query command with error code 2101 ("Unimplemented command").*

- *Transferring a DNSSEC signed domain name to a Registrar who is not using DNSSEC is not possible. In such case, DNSSEC has to be deactivated first.*

Example of a domain transfer command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <transfer op="request">
      <domain:transfer
       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>yourname.ch</domain:name>
        <domain:authInfo>
          <domain:pw>ABCDEF</domain:pw>
        </domain:authInfo>
      </domain:transfer>
    </transfer>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a domain transfer response

```
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <domain:trnData xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>yourname.ch</domain:name>
        <domain:trStatus>serverApproved</domain:trStatus>
        <domain:reID>TEST-REGISTRAR-A</domain:reID>
        <domain:reDate>2007-12-06T16:23:52+01:00</domain:reDate>
        <domain:acID>TEST-REGISTRAR-A</domain:acID>
        <domain:acDate>2007-12-06T16:23:52+01:00</domain:acDate>
      </domain:trnData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20071206.91098.282082</svTRID>
    </trID>
  </response>
</epp>
```

4.2.9 *domain:*update

Updates are made to the domain name with `domain:update`. This command can also be used to generate the transfer code (`<domain:authInfo>`), which is then forwarded to the holder so that the domain name can be transferred.

If the `domain:update` command does not change any entries in the boxes, error code 2308 (“Data management policy violation”) will be returned.

If SWITCH contact IDs are registered at the time the `domain:update` command is submitted (because a transfer has taken place beforehand), the Registrar must enter their own contact IDs with `domain:update`. If only the contact ID for the holder is replaced, the technical contact remains empty. Any technical contact in the box will be deleted.

- `<admin>` *and* `<billing>` *are ignored*
- `<domain:authInfo>`: *the transfer code must comply with the following rules:*
  - *6 – 60 characters (character sets as per Paragraph 3.2)*
  - *No blank spaces, commas or semicolons*

Example of a domain update command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <update>
      <domain:update
       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>yourname.ch</domain:name>
        <domain:add>
          <domain:ns>
            <domain:hostObj>ns.yourns.ch</domain:hostObj>
          </domain:ns>
         <domain:contact
          type="tech">NEWTECHCONTACT</domain:contact>
        </domain:add>
           <domain:rem>
             <domain:contact
              type="tech">OLDTECHCONTACT</domain:contact>
           </domain:rem>
        <domain:chg>
          <domain:authInfo>
            <domain:pw>2BARfoo</domain:pw>
          </domain:authInfo>
        </domain:chg>
      </domain:update>
    </update>
    <clTRID>Test3</clTRID>
  </command>
</epp>
```

DNSSEC Extension

adding DNSSEC data:

```
<extension>
  <secDNS:update
   xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
    <secDNS:add>
      <secDNS:dsData>
        <secDNS:keyTag>16915</secDNS:keyTag>
        <secDNS:alg>13</secDNS:alg>
        <secDNS:digestType>2</secDNS:digestType>
       <secDNS:digest>CAFFEEBABE00D87A0147EFE9877
         AB7335206ABFF6FA730BD6239D65CBAC7E768</secDNS:digest>
        </secDNS:dsData>
      </secDNS:add>
   </secDNS:update>
</extension>
```

replace DNSSEC data (all existing DS records will be replaced):

```
<extension>
  <secDNS:update
  xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
    <secDNS:rem>
      <secDNS:all>true</secDNS:all>
    </secDNS:rem>
    <secDNS:add>
      <secDNS:dsData>
        <secDNS:keyTag>16915</secDNS:keyTag>
        <secDNS:alg>13</secDNS:alg>
        <secDNS:digestType>1</secDNS:digestType>
        <secDNS:digest>CAFFEEBABE00D87A0147EFE9877
         AB7335206ABFF6FA730BD6239D65CBAC7E768</secDNS:digest>
      </secDNS:dsData>
    </secDNS:add>
  </secDNS:update>
</extension>
```

remove DNSSEC data (all DS records that contain enclosed attributes will be deleted):

```
<extension>
  <secDNS:update
  xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
    <secDNS:rem>
      <secDNS:dsData>
        <secDNS:keyTag>16915</secDNS:keyTag>
        <secDNS:alg>13</secDNS:alg>
        <secDNS:digestType>2</secDNS:digestType>
        <secDNS:digest>CAFFEEBABE00D87A0147EFE9877
         AB7335206ABFF6FA730BD6239D65CBAC7E768</secDNS:digest>
      </secDNS:dsData>
    </secDNS:rem>
  </secDNS:update>
</extension>
```

DNSSEC Extension

removing all DNSSEC data:

```
<<extension>
   <secDNS:update xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.1">
     <secDNS:rem>
         <secDNS:all>true</secDNS:all>
    </secDNS:rem>
  </secDNS:update>
 </extension>
```

Example of a domain update command with an RGP restore request

```
<?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">
  <command>
    <update>
      <domain:update xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>yourname.ch</domain:name>
        <domain:chg/>
      </domain:update>
    </update>
    <extension>
     <rgp:update xmlns:rgp="urn:ietf:params:xml:ns:rgp-1.0" xsi:schemaLocation="urn:ietf:params:xml:ns:rgp-1.0
      rgp-1.0.xsd">
      <rgp:restore op="request"/>
     </rgp:update>
    </extension>
  </command>
</epp>
```

or without xsi:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
   <update>
     <domain:update xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
       <domain:name>ugha.ch</domain:name>
       <domain:chg />
     </domain:update>
   </update>
   <extension>
    <rgp:update xmlns:rgp="urn:ietf:params:xml:ns:rgp-1.0">
     <rgp:restore op="request" />
    </rgp:update>
   </extension>
  </command>
</epp>
```

Example of a domain update response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>Test3</clTRID>
      <svTRID>20071207.92654.285546</svTRID>
    </trID>
  </response>
</epp>
```

4.3 Contact commands

4.3.1 General information

In addition to what is set out below, the `contact` commands are described in detail in Reference (3). Only the data permitted by law is published (see Paragraph 3.2). The way in which access to other data and data updating is handled is governed by Reference (06

Contact attribute	mandatory	Update possible	Comment
Id	Yes	No	Server-unique id, ASCII-upper-case letters, numbers and ”-“, supplied by the registrar
Roid	Yes	No	Cnnnnnnnn-SWITCH, supplied by the registry
Status	Yes	No	Linked, ok, serverDeleteProhibited, serverUpdateProhibited
name type=”loc”	Yes	Yes	Last name First name
org type=“loc“	No	Yes	
Street	Yes	Yes	
Street	No	Yes	
Street	No	Yes	
City	Yes	Yes	Maximum Length: 30 signs
Sp	No	Yes	Canton, Federal State, Country
Pc	No	Yes	
Cc	Yes	Yes	ISO.3166.1997
Voice	No	Yes	
Fax	No	Yes	
email	Yes	Yes	
clID	Yes	No	Set by the registry
crID		No	Set by the registry
crDate		No	Set by the registry
upID		No	This is omitted.
upDate		N	This is omitted.
trDate		N	This is omitted.
authinfo			This is ignored. ”not supported“
Disclose			Error 2308

4.3.2 *contact:*check

The `contact:check` command can be used to check whether contacts actually exist. The server replies with registered / not registered. Bulk queries are permitted as per Reference (06) for the purpose specified there and after consultation with SWITCH.

Example of a contact check command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <check>
      <contact:check
       xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
        <contact:id>AVAILABLE-CONTACT-ID</contact:id>
        <contact:id>NOT-AVAILABLE-CONTACT-ID</contact:id>
      </contact:check>
    </check>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a contact check response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <contact:chkData xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
        <contact:cd>
          <contact:id avail="1">AVAILABLE-CONTACT-ID</contact:id>
        </contact:cd>
        <contact:cd>
          <contact:id avail="0">NOT-AVAILABLE-CONTACT-ID</contact:id>
          <contact:reason>In use</contact:reason>
        </contact:cd>
      </contact:chkData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20071207.92655.285552</svTRID>
    </trID>
  </response>
</epp>
```

4.3.3 *contact:*info

The `contact:info` command is used to request information on contacts.

contact:info requests are only answered successfully if one of the following criteria is met:

- the queried contact is managed by the querying registrar (sponsoring client)
- the query contains a valid `<contact:authInfo>` element as follows

Example contact info Command with `<contact:authInfo>` element

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
    <command>
        <info>
            <contact:info xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
                <contact:id>CH-TECH-ID</contact:id>
                <contact:authInfo>
                    <contact:pw roid="D7033341-SWITCH">authinfo_pwd</contact:pw>
                </contact:authInfo>
            </contact:info>
        </info>
        <clTRID>ABC</clTRID>
    </command>
</epp>
```

The attribute roid="D7033341-SWITCH" in the `<contact:pw>` element must be present and refers to a domain name where the queried contact is entered as `<domain:registrant>` or as `<domain:contact>`. (see 4.2.5 domain:info) The value of the `<contact:pw>` element (here `authinfo_pwd`) must be the correct transfer code of the domain name specified by the roid attribute. The corresponding domain name must not be in the RGP phase. If the information in the optional `<contact:authInfo>` is incorrect, 2201 Authorization Error is returned.

Example of a contact info command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <info>
      <contact:info
       xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
        <contact:id>TEST-REGISTRAR-C-5</contact:id>
      </contact:info>
    </info>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a contact info response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <contact:infData xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
        <contact:id>TEST-REGISTRAR-C-5</contact:id>
        <contact:roid>C10577393-SWITCH</contact:roid>
        <contact:status s="ok"/>
        <contact:status s="linked"/>
        <contact:postalInfo type="loc">
          <contact:name>Test User3</contact:name>
          <contact:org>SWITCH</contact:org>
          <contact:addr>
            <contact:street>Test department 3</contact:street>
            <contact:street>Teststrasse 3</contact:street>
            <contact:city>Aarau</contact:city>
            <contact:pc>5000</contact:pc>
            <contact:cc>CH</contact:cc>
          </contact:addr>
        </contact:postalInfo>
        <contact:voice>+41.22222555</contact:voice>
        <contact:email>test2@test1.ch</contact:email>
        <contact:clID>TEST-REGISTRAR-C</contact:clID>
        <contact:crID>TEST-REGISTRAR-C</contact:crID>
        <contact:crDate>2007-12-07T10:42:59+01:00</contact:crDate>
      </contact:infData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20071207.92656.285558</svTRID>
    </trID>
  </response>
</epp>
```

Query about a contact administered by the Registrar who is submitting the query.

4.3.4 *contact:*create

`contact:create` is used to generate contact IDs. Each Registrar has to generate and update their own contact IDs for the domain names they administer. This is particularly important in the case of a transfer, since no updates can be made to a domain name before the Registrar has entered their own contact IDs.

- `<contact:postalInfo type="loc">` *Only the address type* `"loc"` *(localized) is supported. The address type* `"int"` *(international) (7-bit ASCII characters) is not permitted. A command with this address type will be rejected. If attributes with* `type="int"` *and* `type="loc"` *are submitted at the same time, the address type with* `type="int"` *will be ignored.*
- `<contact:street>` *An address can have up to three lines for the street (address) but must contain at least one* `<contact:street>` *element. If there is a PO Box, this should be entered in this element.*
- `<contact:disclose>` *is not supported.*
- `<contact:email>` *The holder's e-mail address must be specified under contacts. The Registrar's e-mail address must not be entered in this box.*
- `<contact:pw>` *is not used by SWITCH and is not saved. Authentication is performed during the EPP login.*
- `<contact:id>` *must be server-unique on the EPP server and contain at least one ASCII upper-case letter. Numbers and hyphens are permitted.*

Example of a contact create command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <contact:create xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
                <contact:id>TEST-REGISTRAR-C-5</contact:id>
                <contact:postalInfo type="loc">
                <contact:name>Test User2</contact:name>
                <contact:org>SWITCH</contact:org>
                   <contact:addr>
                <contact:street>Test department 2</contact:street>
                <contact:street>Teststrasse 2</contact:street>
                <contact:city>Aarau</contact:city>
                <contact:pc>5000</contact:pc>
                <contact:cc>CH</contact:cc>
          </contact:addr>
        </contact:postalInfo>
                <contact:voice>+41.22222555</contact:voice>
                <contact:email>test2@test1.ch</contact:email>
        <contact:authInfo>
          <contact:pw/>
        </contact:authInfo>
            </contact:create>
        </create>
        <clTRID>Test2</clTRID>
    </command>
</epp>
```

Example of a contact create response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <contact:creData xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
        <contact:id>TEST-REGISTRAR-C-5</contact:id>
        <contact:crDate>2007-12-07T11:29:51+01:00
        </contact:crDate>
      </contact:creData>
    </resData>
    <trID>
      <clTRID>Test2</clTRID>
      <svTRID>20071207.944.74886</svTRID>
    </trID>
  </response>
</epp>
```

4.3.5 *contact:*delete

Using `contact:delete` it is possible to delete contact IDs insofar as they are not linked to anything (do not have the status “linked”). Only contact IDs administered by the Registrar themselves can be deleted. The Registrar must delete any of their contact IDs that are no longer in use.

Example of a contact delete command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <delete>
      <contact:delete
       xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
        <contact:id>TEST-REGISTRAR-C-6</contact:id>
      </contact:delete>
    </delete>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a contact delete response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20071207.92657.285565</svTRID>
    </trID>
  </response>
</epp>
```

4.3.6 *contact:*transfer

The `contact:transfer` command is not supported and is rejected with error code 2101.

4.3.7 *contact:*update

`contact:update` is used to update contact IDs. The conditions that apply here are the same as for `contact:create`. The `<contact:addr>` address can only be updated as an entire block.

- `<contact:postalInfo type="loc">` *Only the* `"loc"` *(localized) address type is supported. The* `"int"` *(international) address type (7-bit ASCII characters) is not permitted. A command with this address type will be rejected. If attributes with* `type="int"` *and* `type="loc"` *are sent at the same time, the address type with* `type="int"` *will be ignored.*
- `<contact:street>` *An address can have up to three lines for the street (address) but must contain at least one* `<contact:street>` *element. If there is a PO Box this should be entered in this element.*
- `<contact:disclose>` *is not supported.*
- `<contact:email>` *The holder's e-mail address must be specified under contacts. The Registrar's e-mail address must not appear in this box.*
- `<contact:pw>` *is not used by SWITCH and is not saved. Authentication is performed during the EPP login.*
- `<contact:id>` *must be server-unique on the EPP server and contain at least one ASCII upper-case letter. Numbers and hyphens are permitted.*

Example of a contact update command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <update>
      <contact:update
       xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
        <contact:id>TEST-REGISTRAR-C-5</contact:id>
        <contact:chg>
          <contact:postalInfo type="loc">
            <contact:org/>
            <contact:addr>
              <contact:street>124 Example Dr.</contact:street>
              <contact:street>Suite 200</contact:street>
              <contact:city>Dulles</contact:city>
              <contact:sp>VA</contact:sp>
              <contact:pc>20166-6503</contact:pc>
              <contact:cc>US</contact:cc>
            </contact:addr>
          </contact:postalInfo>
          <contact:voice>+1.7034444444</contact:voice>
          <contact:fax/>
          <contact:authInfo>
            <contact:pw/>
          </contact:authInfo>
        </contact:chg>
      </contact:update>
    </update>
    <clTRID>freechoice</clTRID>
  </command>
</epp>
```

Example of a contact update response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>freechoice</clTRID>
      <svTRID>20071207.92690.285728</svTRID>
    </trID>
  </response>
</epp>
```

4.4 Host commands

4.4.1 General information

In addition to what is set out below, all the `host` commands are described in detail in Reference (4).
A distinction is drawn between the following hosts (name servers):

Valid internal name server under .ch/.li	**Invalid internal name server under .ch/.li**	**External name server not under .ch/.li**
Ends in .ch or .li, superordinate domain name is registered	Ends in .ch or .li, superordinate domain name is not registered	Name server that does not end in .ch or .li
ns1.iamregistered.ch	ns1.iam**not**registered.ch	ns1.yourname.**com**
Domain name iamregistered.ch is registered	Domain name iamnotregistered.ch is not registered	Irrespective of whether yourname.com is registered or not

Valid internal name servers under .ch/.li belong to the holder of the superordinate domain name and are administered by their Registrar. These are transferred together with the domain name.

Invalid internal name servers under .ch/.li are administered by SWITCH insofar as the superordinate domain name is not registered. It is permitted to register an invalid internal name server. An invalid internal name server may only be allocated to the superordinate domain name, thus making it into a valid internal name server.

If a domain name (yourname.ch) is deleted, the subordinate name servers (ns1.yourname.ch) continue to exist. If the domain name is re-registered again later, these name servers automatically belong to the holder of the new domain name.

External name servers such as ns1.yourname.com, for example, can be registered and belong to the registry.

host attribute	mandatory	update possible	Comment
Name	Yes	No	
Roid	Yes	No	Hnnnnnnnn-SWITCH, supplied by the registry
Status	Yes	No	
clID	Yes	No	Set by the registry
crID			“NOT SUPPORTED”
crDate		No	Set by the registry
upID			This is omitted.
update		No	This is omitted.
trDate		No	This is omitted.

4.4.2 *host:*check

With `host:check` it is possible to check whether name servers exist. The server will reply with registered / not registered. Bulk queries are only permitted to a limited extent, as per Reference (06).

- *A maximum of 10 hosts can be queried at a time with* `host:check`*, otherwise the server will answer with error code 2308 (“Data management policy violation”).*

Example of a host check command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <check>
      <host:check
       xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:name>merapi.switch.ch</host:name>
        <host:name>ns2.example.com</host:name>
      </host:check>
    </check>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a host check response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <host:chkData xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:cd>
          <host:name avail="0">merapi.switch.ch</host:name>
          <host:reason>In use</host:reason>
        </host:cd>
        <host:cd>
          <host:name avail="1">ns2.example.com</host:name>
        </host:cd>
      </host:chkData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20071207.92693.285743</svTRID>
    </trID>
  </response>
</epp>
```

4.4.3 *host:*info

`host:info` is used to request information on name servers. The `<host:clID>`, `<host:crID>` and `<host:crDate>` boxes are only disclosed for name servers administered by the Registrar themselves.

Example of a host info command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <info>
      <host:info
       xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:name>merapi.switch.ch</host:name>
      </host:info>
    </info>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a host check response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <host:infData xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:name>merapi.switch.ch</host:name>
        <host:roid>H5734-SWITCH</host:roid>
        <host:status s="ok"/>
        <host:status s="linked"/>
        <host:addr ip="v4">130.59.211.10
        </host:addr>
        <host:addr ip="v6">2001:620:0:0:0:0:0:5</host:addr>
        <host:clID>SWITCH-REGISTRAR</host:clID>
        <host:crID>NOT SUPPORTED</host:crID>
        <host:crDate>1999-11-24T14:48:50+01:00</host:crDate>
      </host:infData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20071207.92693.285744</svTRID>
    </trID>
  </response>
</epp>
```

4.4.4 *host:*create

The `host:create` command is used to register name servers. The authorisations listed under Paragraph 4.4.1 above apply. Anyone is entitled to create external and invalid internal name servers. Valid internal name servers can only be set up by the Registrar who administers the superordinate domain name.

Up to 20 IP addresses per host and 20 hosts per domain name allowed.

Example of a host create command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <host:create
       xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:name>ns1.yourname.ch</host:name>
        <host:addr ip="v4">192.0.2.2</host:addr>
        <host:addr ip="v4">192.0.2.29</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>
```

Example of a host create response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <resData>
      <host:creData xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:name>ns1.yourname.ch</host:name>
        <host:crDate>2007-12-07T14:40:04+01:00</host:crDate>
      </host:creData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20071207.92694.285749</svTRID>
    </trID>
  </response>
</epp>
```

4.4.5 *host:*transfer

No provision is made for this command in Reference (03), and it will be rejected with error code 2000. Valid internal name servers belong to the holder of the corresponding domain name. The name servers are transferred together with the domain name.

4.4.6 *host:*delete

The `host:delete` command is used to delete name servers. Name servers may only be deleted if they are no longer referenced in the registration system (if no domain name points to the name server).

Example of a host delete command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <delete>
      <host:delete
       xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:name>ns1.yourname.ch</host:name>
      </host:delete>
    </delete>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a host delete response

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg lang="en">Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20071210.94223.292958</svTRID>
    </trID>
  </response>
</epp>
```

4.4.7 *host:*update

Example of a host update command

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <update>
      <host:update
       xmlns:host="urn:ietf:params:xml:ns:host-1.0">
        <host:name>ns1.yourns.ch</host:name>
        <host:add>
          <host:addr ip="v4">192.0.2.22</host:addr>
        </host:add>
        <host:rem>
          <host:addr ip="v6">1080:0:0:0:8:800:200C:417A</host:addr>
        </host:rem>
      </host:update>
    </update>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example of a host update response

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>20100112.436723.6100466</svTRID>
    </trID>
  </response>
</epp>
```

The name of a name server cannot be updated via the EPP interface.

5 Testing

5.1 Test system

SWITCH can activate access to the EPP test server (**epp-test.switch.ch**) for test purposes. The availability of the EPP test server is lower than that of the productive EPP servers. As a rule, the EPP test server will have the same software version installed on it as the productive EPP server. Prior to new software releases, the new version will generally be made available on the EPP test server 3 days before it is installed on the productive EPP server. Exceptions to this rule apply in the case of urgent bug fixes, security measures or major technical adjustments. In such case, the one-week-period changes or a new transition period will be set.

5.1.1 Requirements for the test system

- For the test system, the registrar receives two individual Client Identifier clIDs (TEST-XXXXXXXXX, TEST-XXXXXXXXX) and the passwords.
- These cllDs can be used by the registrar during the entire contract period.
- To set up the test system, SWITCH needs at least one IP address.
- **Connection**
  Host: epp-test.switch.ch
  Port: 7001

5.2 Test procedure

Before SWITCH activates the EPP interface to the productive environment (epp.nic.ch) for a Registrar to use, the test procedure described below must have been successfully completed. A test period must be agreed on with SWITCH for conducting the test. Only one registrar can take the test procedure at a time. In addition to this, the provisions set out in Reference (6) apply.

The registrar receives a separate clID and password from SWITCH. These credentials are only valid for the test procedure and cannot be used for the test system described above.

5.2.1 Requirements for the test procedure

- The registrar has tested extensively on the test system (epp-test.switch.ch).
- For the test procedure the registrar receives a clID and a password.
- This cllD is only valid for the test procedure and not for the test system.
- To set up the test procedure, SWITCH needs at least one IP address.
- **Connection**
  Host: epp-test.switch.ch
  Port: 7001

5.2.2 Process for the test procedure

The individual commands are to be worked through in the order shown. After each command, the response from the EPP server must tally with the expected result.

Step1 - Login and change password

Description / Tag	Value	Comment
clID	test-registrar-a	Test-account client-identifier
Pw	**********	Initial passwort
newPW	**********	Password chosen by the Registrar, observing the password rules in Reference (6)

Expected result: “Command completed successfully”.

Step 2 – Poll and confirm messages

Description / Tag	Value	Comment
poll op="req"		Extract Msg_Id

Expected result: “Command completed successfully; ack to dequeue”.

Step 3 - Transfer domain 1

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	
domain:pw	'my&p$w#d22.'	Transfer Code without the single quotes (apostrophe)

Expected result: “Command completed successfully”.

Step 4 – Acknowledge messages

Description / Tag	Value	Comment
poll op="ack" msgID="nn”		Extract msgID from the poll op=”req” and insert at nn

Expected result: “Command completed successfully”.

Step 5 - Check contact 1

Description / Tag	Value	Comment
contact:id	TEST-CONTACT-1	

Expected result: “Command completed successfully. The contact:id is available.”

Step 6 - Create contact 1

Description / Tag	Value	Comment
contact:id	TEST-CONTACT-1	
contact:name	Lastname Firstname	
contact:org	Organisation	
contact:street	Testabteilung	
contact:street	Teststrasse 999	
contact:city	Bern	
contact:pc	3001	
contact:cc	CH	
contact:voice	+41.335555555	
contact:email	test1@yourdomain.ch	

Expected result: “Command completed successfully”.

Step 7 - Create name server 1

Description / Tag	Value	Comment
host:name	ns3.test-registrar-a-domain-1.ch	
host:addr ip="v4"	240.1.1.1	

Expected result: “Command completed successfully”.

Step 8 - Info domain 1

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	

Expected result: “Command completed successfully”. The response contains the contact:id for the technical contact for the next step.

Step 9 - Info Tech-Contact

Description / Tag	Value	Comment
contact:id		Value from domain:contact type="tech"

Expected result: “Command completed successfully”.

Step 10 - Update domain 1 (complete transfer)

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	
domain:add domain:ns	ns3.test-registrar-a-domain-1.ch	
domain:add domain:contact type="tech"	TEST-CONTACT-1, or with another newly created contact	At this time is no domain:rem needed.
domain:rem domain:ns	ns1.test-registrar-a-domain-1.ch	
domain:chg domain:registrant	TEST-CONTACT-1	

Expected result: “Command completed successfully”.

Step 11 - Create contact 2

Description / Tag	Value	Comment
contact:id	TEST-CONTACT-2	
contact:name	Lastname2 Firstname2	
contact:org	Organisation2	
contact:street	Testdivision	
contact:street	Teststreet 999	
contact:city	Bern	
contact:pc	3001	
contact:cc	CH	
contact:voice	+41.335555555	
contact:email	test2@yourdomain.ch	

Expected result: “Command completed successfully”.

Step 12 - Delete domain 1

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	

Expected result: “Command completed successfully”.

Step 13 - Restore domain 1

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	

Expected result: “Command completed successfully”.

Step 14 - Update domain 1

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	
domain:chg domain:registrant	TEST-CONTACT-2	

Expected result: “Command completed successfully”.

Step 15 - Delete contact 1

Description / Tag	Value	Comment
contact:id	TEST-CONTACT-1	

Expected result: “Command completed successfully”.

Step 16 - Delete name server 1

Description / Tag	Value	Comment
host:name	ns1.test-registrar-a-domain-1.ch	

Expected result: “Command completed successfully”.

Step 17 - Check domain 2

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-2.ch	

Expected result: “Command completed successfully”. The domain name is available.

Step 18 - Create domain 2

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-2.ch	
domain:hostObj	ns3.test-registrar-a-domain-1.ch	
domain:registrant	TEST-CONTACT-2	

Expected result: “Command completed successfully”.

Step 19 - Create name server 2 subordinate of domain 2

Description / Tag	Value	Comment
host:name	ns1.test-registrar-a-domain-2.ch	
host:addr ip="v4"	240.1.1.1	

Expected result: “Command completed successfully”.

Step 20 - Update domain 1 with ns 2

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	
domain:add domain:ns	ns1.test-registrar-a-domain-2.ch	

Expected result: “Command completed successfully”.

Step 21 - Update contact 2 (All the values not mentioned should remain unchanged)

Description / Tag	Value	Comment
contact:id	TEST-CONTACT-2	
contact:chg contact:org		Empty
contact:chg contact:addr contact:street	New Division	
contact:chg contact:addr contact:street	Teststreet 999	previous value

Expected result: “Command completed successfully”.

Step 22 - Update domain 2 with authinfo

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-2.ch	
domain:chg domain:authInfo domain:pw	2BARfoo	

Expected result: “Command completed successfully”.

Step 23 - Update domain 1 with DNSSEC data

Steps 22 to 24 are only necessary if the Registrar will be using DNSSEC. Logging in with :secDNS extension is required and that all the steps prior have been accomplished-

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	
secDNS:add secDNS:dsData secDNS:keyTag secDNS:alg secDNS:digestType secDNS:digest	 12346 13 2 CAFFEEBABE00D87A0147EFE9877AB7335206ABFF6FA730BD6239D65CBAC7E768	

Expected result: “Command completed successfully”.

Step 24 – Check update with domain:info

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	
domain:info		

Expected result: “Command completed successfully”.

Step 25 – Replace DNSSEC data

Description / Tag	Value	Comment
domain:name	test-registrar-a-domain-1.ch	
secDNS:rem secDNS:dsData secDNS: keyTag secDNS:alg secDNS:digestType secDNS:digest	 12346 13 2 CAFFEEBABE00D87A0147EF E9877AB7335206ABFF6FA73 0BD6239D65CBAC7E768	
secDNS:add secDNS:dsData secDNS:keyTag secDNS:alg secDNS:digestType **secDNS:digest**	 44475 14 2 AABBCCBABE00D87A0147EF E9877AB7335206ABFF6FA73 0BD6239D65CBAC7E768	

Expected result: “Command completed successfully”.

Step 26 - Logout

Expected result: successfully completed, ending session.

6 Further information

6.1 Web interface for EPP users

Account statements and other information is available via the web interface. Domain names cannot be administered via the web interface.

Contact: **registrar@nic.ch**

Appendix A: Abbreviations / Terms

Term	Explanation
AuthInfo	The transfer code that is required to transfer a domain name.
ccTLDs	country-code-Top-Level domain names
Command	Command that can be sent by the EPP client in order to trigger a specific action on the EPP server. The actions are allocated to an object (domain name, name server and contact).
Contact ID	Handle / object reference
DNSSEC	Domain Name System Security Extension
EPP	Extensible Provisioning Protocol
EPP User	Registrars who communicate with SWITCH via the EPP interface.
EPP Interface	Interface based on the EPP protocol.
External name server	A name server that does not belong to the ccTLD administered by the registration system. At SWITCH, this is a name server that does not end with .ch or .li
Internal name server	A name server that belongs to the ccTLD administered by the registration system. At SWITCH, this is a name server that ends with .ch or .li.
Valid internal name server	An internal name server of a registered superordinate domain name.
Invalid internal name server	An internal name server of a non-registered superordinate domain name.
Registrar	An Internet Service Provider with a Registrar Agreement signed by SWITCH
Registrant	End customer who owns a domain name.
Registrar	International designation for Registrars.
Registry	Organisation that acts as the registry for second-level domain names.
Transfer	Transferring the administrative rights to a domain name to another Registrar.
TLS	Transport Layer Security; security protocol for internet connections employing a certificate.
Holder transfer	The domain name is allocated to a different holder, i.e. to another natural or legal person.