Media Cloud Indonesia - EPP Integration Guide

1. Management Summary

The EPP Protocol (Extensible Provisioning Protocol) is an international standard for managing domain names. It enables secure and continuous communication between Resellers and Media Cloud Indonesia's registry system, ensuring seamless domain administration.

This document provides all the necessary information for Resellers to connect their EPP client to Media Cloud Indonesia EPP Server. Our EPP implementation follows international standards, allowing resellers to integrate domain management efficiently into their existing systems.

2. Introduction

Media Cloud Indonesia is responsible for registering and administering domain names within its system. One of its primary tasks is facilitating domain name registration as an accredited reseller of PANDI (.id). Media Cloud Indonesia provides a platform for resellers to manage and sell domain names efficiently under-supported Top-Level Domains (TLDs), which will be referred to simply as "domain names" in this document.

2.1 Production and OT&E Environments

Registrars should add SSL certificate and IP Whitelists in each of these environments separately. In each environment, you can add your IP addresses for protected access via the *IP Whitelists* Menu. You can also add SSL Certificates signed by trusted Certificate Authorities by clicking on the *SSL Certificates* Menu.

Production

EPP Hostname	reseller-epp.mediacloud.id
EPP Port	700
Dashboard	reseller.mediacloud.id

OTE

EPP Hostname	ote-reseller-epp.mediacloud.id
EPP Port	700
Dashboard	ote-reseller.mediacloud.id

Credentials

To login to the EPP, you need to use your **Reseller Client ID** as **username** and the EPP **password** will be sent via **email**. You could also change the password in the dashboard.

2.2 EPP standard + legal fundamentals

Reference	Document
[01]	RFC 5730: Extensible Provisioning Protocol (EPP)

[02]	RFC 5731: Extensible Provisioning Protocol (EPP) Domain Name Mapping
[03]	RFC 5732: Extensible Provisioning Protocol (EPP) Host Mapping
[04]	RFC 5733: Extensible Provisioning Protocol (EPP) Contact Mapping
[05]	RFC 5734: Extensible Provisioning Protocol (EPP) Transport over TCP
[06]	RFC 5910: Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)

2.2 The Use of Conditions

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

- A signed and valid Reseller Agreement
- An IP address and SSL Certificate for your EPP client that Media Cloud Indonesia has activated
- A completed test run (see Paragraph 5.3)

3. Using the EPP interface

An EPP client with a secure TLS connection is essential for accessing the EPP interface provided by Media Cloud Indonesia. Only TLS versions 1.2 and above are supported. Before establishing a TLS connection, the reseller's IP address and SSL certificate must be registered via Media Cloud Indonesia Reseller Dashboard. Once the TLS connection is established, only the EPP `login` and `hello` commands are permitted. Other EPP commands can only be executed after a successful login. The server will also send a greeting to the user after successful connection.

`greeting` example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <greeting>
        <svID>MCI</svID>
        <svDate>2025-04-11T22:42:24.75921+07:00</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>
            </svcExtension>
        </svcMenu>
        <dcp>
            <access>
                <all></all>
            </access>
            <statement>
                <purpose>
                    <prov></prov>
                </purpose>
                <recipient>
                    <ours></ours>
                    <public></public>
                </recipient>
```

```
                <retention>
                    <stated></stated>
                </retention>
            </statement>
        </dcp>
    </greeting>
</epp>
```

Media Cloud Indonesia does not provide a standard EPP client or EPP library. Resellers are responsible for using an EPP client that complies with the referenced standards.

In addition to standard RFC requirements, an EPP client must be configured by this Manual and the Reseller Agreement, including its annexes. Communication between the EPP client and the EPP server is conducted using XML-formatted commands. The server will send immediate responses to each command received. If multiple commands are sent in succession (pipelining), they will be processed sequentially, ensuring that a response is returned for each command before processing the next.

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 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 the user receives notifications.

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

3.2 Media Cloud Indonesia-specific general points

Under the terms of Reference (01), all EPP commands are **atomic**, meaning they are either fully processed or completely rejected. If the same EPP command is submitted twice, both submissions will be processed independently. If a domain name has already been registered during the first submission, an error message will be returned upon the second submission with the same domain name.

In general, any non-mandatory information submitted by the EPP client that is not required will be ignored.

1. Each Reseller is allowed to maintain only one active EPP session at a time.
2. A single IP address can only be associated with one active EPP client connection.
3. The domain name transfer code must consist only of alphanumeric characters (e.g., dh2J3EfX).

3.3 Session ended

An EPP session can be terminated by the server under the following conditions:

1. The session remains inactive for more 5 **minutes** (**session timeout**)
2. The session remains connected for more than 1 hour.
3. The maximum number of **unsuccessful EPP commands** has been reached.
4. There is **scheduled or unscheduled maintenance** on the EPP server.

4. EPP commands

Each reseller has a maximum limit on the number of EPP commands that can be executed per minute. Under normal usage conditions, this limit is set to 100 EPP commands per minute. However, based on system load and operational requirements, Media Cloud Indonesia

reserves the right to adjust this limit dynamically. If the maximum threshold is reached, the server will temporarily delay processing any additional EPP requests until capacity becomes available.

4.1 Protocol commands

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

4.1.1 login

An EPP session is set up using a) a TLS connection to the EPP server and b) a subsequent successful login with the EPP login command according to 3.2, except for 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 will send the EPP response after the `login` command. If response contains code `1000` means that the EPP session has been successfully established.

After the login is successful 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, the password must contain lowercase and uppercase letters, at least one digit, and one special character. Particular attention should be paid to the provisions of the GTCs when it comes to the careful selection of passwords and keeping passwords safe and confidential.

`login` command example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"
xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <command>
    <login>
      <clID>foobar</clID>
      <pw>password</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>
        <svcExtension>
          <extURI>urn:ietf:params:xml:ns:secDNS-1.1</extURI>
          <extURI>urn:ietf:params:xml:ns:rgp-1.0</extURI>
        </svcExtension>
      </svcs>
    </login>
    <clTRID>LOGIN-4oQ8zbwMM18ShXJZtjBWRP</clTRID>
  </command>
</epp>

```

`login` response example

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
```

```
    <response>
        <result code="1000">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>LOGIN-4oQ8zbwMM18ShXJZtjBWRP</clTRID>
            <svTRID>0195b3f6-b57c-7079-bda2-76e3fa3fd2f5</svTRID>
        </trID>
    </response>
</epp>

```

4.1.2 Logout

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

`logout` command example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"
xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <command>
    <logout/>
    <clTRID>LOGOUT-9EQhKsQDuspYMH5scguBBf</clTRID>
  </command>
</epp>

```

`logout` response example

```
<?xml version="1.0" encoding="UTF-8"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
    <response>
        <result code="1000">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>LOGOUT-9EQhKsQDuspYMH5scguBBf</clTRID>
            <svTRID>0195b3f6-b5c4-788b-bfe0-6248ef66dc52</svTRID>
        </trID>
    </response>
</epp>

```

4.1.3 hello

The EPP client can issue the hello command to verify if the EPP session remains active. The EPP server responds to the hello command with a greeting message.

`hello` command example

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

```

`hello` response example

```
<?xml version="1.0" encoding="UTF-8" ?>
<epp
    xmlns="urn:ietf:params:xml:ns:epp-1.0">
    <greeting>
        <svID>Media Cloud Indonesia</svID>
        <svDate>2025-04-11T09:15:48.445322+07:00</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>
            </svcExtension>
        </svcMenu>
        <dcp>
            <access>
                <all></all>
            </access>
            <statement>
                <purpose>
                    <prov></prov>
                </purpose>
                <recipient>
                    <ours></ours>
                    <public></public>
                </recipient>
                <retention>
                    <stated></stated>
                </retention>
            </statement>
        </dcp>
    </greeting>
</epp>
```

4.1.4 poll

Messages in the poll queue must be fetched one by one and confirmed. Once the first message is confirmed, the next message can be retrieved, and so on. Confirmed messages are automatically removed from the poll queue.

The command includes the element with the “op” parameter, which accepts the following values:

1. req – retrieves the first message from the queue
2. ack– confirms receipt of the message. In this case, the element must also include the “msgID” parameter containing a unique message identifier, where the value matches the "id" attribute taken from the element in the response to the command.

`poll` command with `req` example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"
xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <command>
    <poll op="req"/>
    <clTRID>POLL_REQ-mci-s8Cu2CF8RaXPkfzF4QfJgB</clTRID>
  </command>
</epp>

```

`poll` response with code `req` when there is no pending polls example

```
<?xml version="1.0" encoding="UTF-8" ?>
<epp
    xmlns="urn:ietf:params:xml:ns:epp-1.0">
    <response>
        <result code="1000">
            <msg>Command completed successfully</msg>
        </result>
        <msgQ count="0" id="0"></msgQ>
        <trID>
            <clTRID>POLL_REQ-mci-s8Cu2CF8RaXPkfzF4QfJgB</clTRID>
            <svTRID>019622a5-8883-79c4-8bd7-99a901f15508</svTRID>
        </trID>
    </response>
</epp>
```

`poll` response with code `req` when there is pending polls example

```
<?xml version="1.0" encoding="UTF-8" ?>
<epp
    xmlns="urn:ietf:params:xml:ns:epp-1.0">
    <response>
        <result code="1000">
            <msg>Command completed successfully</msg>
        </result>
        <msgQ count="25" id="10">
        <qDate>2025-03-11T21:26:01.029+07:00</qDate>
        </msgQ>
        <resData>
            <domain:trnData
                xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
                <domain:name>domain.id</domain:name>
                <domain:trStatus>pending</domain:trStatus>
                <domain:reID>mci</domain:reID>
                <domain:reDate>2025-03-11T21:26:01.029+07:00</domain:reDate>
                <domain:acID>mci</domain:acID>
                <domain:acDate>2025-03-11T22:19:01.156+07:00</domain:acDate>
            </domain:trnData>
        </resData>
        <trID>
            <clTRID>POLL_REQ-kBxVEX9YuU2EHFrXRdVXA5</clTRID>
            <svTRID>019622b5-f70a-771c-809d-ea6533ac70fd</svTRID>
        </trID>
    </response>
</epp>
```

`poll` command with code `ack` example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"
xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <command>
    <poll msgID="3" op="ack"/>
    <clTRID>POLL_REQ-s8Cu2CF8RaXPkfzF4QfJgB</clTRID>
  </command>
</epp>
```

`poll` command with code `ack` example

```
<?xml version="1.0" encoding="UTF-8" ?>
<epp
    xmlns="urn:ietf:params:xml:ns:epp-1.0">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <msgQ count="24" id="10"></msgQ>
        <trID>
            <clTRID>POLL_ACK-aAAWqtPfJuPDSvBQUEx5w1</clTRID>
            <svTRID>019622ba-fa2d-718e-aa1b-2f4ec9ccda99</svTRID>
        </trID>
    </response>
</epp>
```

4.1.4.1 Domain Transfer Completed message

This message is placed in the Registrar's poll queue on the date a domain transfer has been finalized.

Attribute Comment `domain:name` is the domain name, the attribute `domain:trstatus` with status `clientApproved` means the transfer is approved by client. The attribute `domain:reID` is the reseller id of the requester `domain:reDate` is the date of request `domain:acID` is the reseller id of the approver. The attribute `domain:acDate` is the date when an action happened.

4.2 Domain commands

4.2.1 General information

1. The domain name holder is stored in the `domain:registrant` field.
2. The domain name transfer code can only be provided to the domain name holder.

4.2.2 DNSSEC Extension

DNSSEC is optional and used to sign domain names.

DNSSEC Attribute	M	Comment
alg	Y	algorithm (currently 3, 5, 6, 7, 8, 10, 12, 13, 14)
digestType	Y	2 Notes: for now, only accepting type 2 (SHA-256)
digest	Y	max. 96 characters
flags	N	256 or 257
protocol	N	3
keyTag	Y	0-65535 value range
pubKey (DNSKEY)	N	max. 4096 bits

M = Mandatory

Y = Yes

N = No

- The registrar no longer needs to be activated separately to use DNSSEC. When logging in, **DNSSEC** (`urn:ietf:params:xml:ns:secDNS-1.1`) should only be specified if the registrar supports DNSSEC via EPP.
- Media Cloud Indonesia does not perform delegation checks or verify the accessibility of signed domain names.
- **keyData** entries are optional and will be stored in the database if provided.
- If **keyData** is used, the attributes **flags**, **protocol**, **alg**, and **pubKey** are mandatory.
- A maximum of 20 DNSSEC entries is allowed per domain name.

4.2.3 RGP Extension

RGP is optional and can be used to restore deleted domain names.

- RGP must be specified when logging in using `urn:ietf:params:xml:ns:rgp-1.0`.
- The registrar must submit the query for its domain name. The `domain:info` command can be used to check whether the domain name is in the redemption period.
- There is no **pendingRestore** RGP status. If the restore command is successful, the domain name's EPP status will change to **OK**.
- The optional `rgp:report` element inside the `rgp:restore` element will be ignored.

4.2.4 domain:check

The `domain:check` command can be used to verify the availability of domain names. The server will respond with **available “1”** or **not available “0”.**

`domain:check` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <check
            xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
            <domain:check>
                <domain:name>hello.id</domain:name>
                <domain:name>123321123.id</domain:name>
            </domain:check>
        </check>
        <clTRID>DOMAIN_CHECK-kfRdhUom3RkqQndFjjwi8n</clTRID>
    </command>
</epp>
```

`domain:check` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <domain:chkData
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:cd>
                    <domain:name avail=\"0\">hello.id</domain:name>
                    <domain:reason>In Use</domain:reason>
                </domain:cd>
                <domain:cd>
```

```
                    <domain:name avail=\"1\">123321123.id</domain:name>
                </domain:cd>
            </domain:chkData>
        </resData>
        <trID>
            <clTRID>DOMAIN_CHECK-kfRdhUom3RkqQndFjjwi8n</clTRID>
            <svTRID>0196278b-ca44-7a93-8acd-b0217e009062</svTRID>
        </trID>
    </response>
</epp>
```

4.2.5 domain:info

The `domain:info` command is used to retrieve information about a domain name (see Paragraph 3.2). In addition to standard data, the `<domain:clID>` field contains the ID of the **sponsoring client**, which refers to the current registrar's ID. The `<domain:authInfo>` is only disclosed for domain names managed by the registrar submitting the query.

`domain:info` command example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"
xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd">
  <command>
    <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">example.se</domain:name>
      </domain:info>
    </info>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>

```

`domain:info` response example with no auth code

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <domain:infData
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>hello123.id</domain:name>
                <domain:roid>MCI-DO9833507</domain:roid>
                <domain:status s=\"serverTransferProhibited\"></domain:status>
                <domain:status s=\"renewPeriod\"></domain:status>
                <domain:registrant>ID69uaddNO</domain:registrant>
                <domain:contact type=\"admin\">ID69uaddNO</domain:contact>
                <domain:contact type=\"tech\">ID69uaddNO</domain:contact>
                <domain:contact type=\"billing\">ID69uaddNO</domain:contact>
                <domain:ns>
                    <domain:hostObj>ns1.afraid.org</domain:hostObj>
                    <domain:hostObj>ns2.afraid.org</domain:hostObj>
                </domain:ns>
                <domain:clID>mci</domain:clID>
```

```
                <domain:crID>admin-mc1</domain:crID>
                <domain:crDate>2024-08-01T10:12:45Z</domain:crDate>
                <domain:upID>lifecycle-executor</domain:upID>
                <domain:upDate>2024-10-30T13:06:18Z</domain:upDate>
                <domain:exDate>2026-08-01T23:59:59Z</domain:exDate>
            </domain:infData>
        </resData>
        <trID>
            <clTRID>DOMAIN_INFO-c9r4eh6mKb8R1NAzqkm2Cu</clTRID>
            <svTRID>019622d5-4e9d-7497-86b9-5c2b89a33230</svTRID>
        </trID>
    </response>
</epp>
```

`domain:info` response example with auth code

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\n
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <domain:infData
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>hello.id</domain:name>
                <domain:roid>MCI_GaLSeispS7LKw8v88C7L5q-40815657</domain:roid>
                <domain:status s=\"addPeriod\"></domain:status>
                <domain:status s=\"serverTransferProhibited\"></domain:status>
                <domain:registrant>eNkdlGUnpZJ4qi6m</domain:registrant>
                <domain:contact type=\"tech\">eNkdlGUnpZJ4qi6m</domain:contact>
                <domain:contact type=\"admin\">eNkdlGUnpZJ4qi6m</domain:contact>
                <domain:contact type=\"billing\">eNkdlGUnpZJ4qi6m</domain:contact>
                <domain:ns>
                    <domain:hostObj>ns1.afraid.org</domain:hostObj>
                    <domain:hostObj>ns1.afraid.org</domain:hostObj>
                </domain:ns>
                <domain:clID>kOzCvNAFaO4JxBz1</domain:clID>
                <domain:crID>kOzCvNAFaO4JxBz1</domain:crID>
                <domain:crDate>2025-03-08T05:12:52.07Z</domain:crDate>
                <domain:exDate>2026-03-08T23:59:59Z</domain:exDate>
                <domain:authInfo>
                    <domain:pw>
                        <![CDATA[Dz4Ztdxl]]>
                    </domain:pw>
                </domain:authInfo>
            </domain:infData>
        </resData>
        <trID>
            <clTRID>DOMAIN_INFO-wtnv26Gfem5v2BB6vMDtc1</clTRID>
            <svTRID>019622d8-8695-7bb0-9ef4-36265af6b081</svTRID>
        </trID>
    </response>
</epp>
```

4.2.6 domain:create

The `domain:create` command is used to register domain names. Before performing this action, all contacts must be created if they do not already exist. Name servers must also be registered in advance to be assigned to the domain name.

`domain:create` command example with DNS Sec

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <create>
            <domain:create
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>hello123456.id</domain:name>
                <domain:period unit=\"y\">1</domain:period>
                <domain:ns>
                    <domain:hostObj>ns1.afraid.org</domain:hostObj>
                    <domain:hostObj>ns2.afraid.org</domain:hostObj>
                </domain:ns>
                <domain:registrant>Do5o6kn6Ze36ZneE</domain:registrant>
                <domain:contact type=\"tech\">Do5o6kn6Ze36ZneE</domain:contact>
                <domain:contact type=\"admin\">Do5o6kn6Ze36ZneE</domain:contact>
                <domain:contact type=\"billing\">Do5o6kn6Ze36ZneE</domain:contact>
                <domain:authInfo>
                    <domain:pw/>
                </domain:authInfo>
            </domain:create>
        </create>
        <extension>
            <secDNS:create
                xmlns:secDNS=\"urn:ietf:params:xml:ns:secDNS-1.1\">
                <secDNS:dsData>
                    <secDNS:keyTag>63899</secDNS:keyTag>
                    <secDNS:alg>13</secDNS:alg>
                    <secDNS:digestType>2</secDNS:digestType>

<secDNS:digest>D651C9E29E379E3285C23F9A2BC25E5418943CCD4827294B00DCA7A12517E82F</secDNS:digest>
                </secDNS:dsData>
            </secDNS:create>
        </extension>
        <clTRID>DOMAIN_CREATE-kxP17YeS1e4ammJ7J3WKQq</clTRID>
    </command>
</epp>\n
```

`domain:create` a response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <domain:creData
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
```

```
                <domain:name>hello123456.id</domain:name>
                <domain:crDate>2024-03-11T15:15:56Z</domain:crDate>
                <domain:exDate>2025-03-11T23:59:59Z</domain:exDate>
            </domain:creData>
        </resData>
        <trID>
            <clTRID>DOMAIN_CREATE-kxP17YeS1e4ammJ7J3WKQq</clTRID>
            <svTRID>0196256b-bac9-7dbf-9116-5fc7e94856f7</svTRID>
        </trID>
    </response>
</epp>
```

4.2.7 domain:delete

The `domain:delete` command is used to delete domain names. If the domain name has the **addPeriod** status, the deletion takes effect immediately, and the domain name can only be re-registered. Otherwise, the domain name will enter the **redemptionPeriod** status.

An authorized registrar can delete a domain name at any time, even if subordinate name servers (subordinate hosts) exist. These name servers will remain registered but can no longer be used (see Paragraph 4.4).

`domain:delete` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <delete
            xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
            <domain:delete>
                <domain:name>deleted.id</domain:name>
            </domain:delete>
        </delete>
        <clTRID>DOMAIN_DELETE-2KiSwiqRLuaaNNZzb5rRMY</clTRID>
    </command>
</epp>
```

`domain:delete` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>DOMAIN_DELETE-2KiSwiqRLuaaNNZzb5rRMY</clTRID>
            <svTRID>01962fa9-15d7-7635-805c-a630d33b4fbf</svTRID>
        </trID>
    </response>
</epp>
```

4.2.8 domain:transfer

The domain name holder may manage their domain name through a registrar. The `domain:transfer` command is used to transfer domain names. If subordinate name servers exist for the domain name, they will be transferred along with the domain name. Transfer requests will be automatically approved if the losing registrar does not respond (either accept or reject) within 5 days.

Once the transfer is successful, the `<serverTransferProhibited>` status will be assigned, preventing further transfers to another registrar. This status remains active for 60 days and will be automatically removed afterward. Before making further updates, the new registrar must use the `domain:update` command to assign a contact ID that they have created for the holder. Until this is done, only `check`, `info`, `delete`, and `transfer` commands are allowed.

`domain:transfer` command with operation `request` example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <transfer op=\"request\">
            <domain:transfer
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>trannsfer.id</domain:name>
                <domain:period unit=\"y\">1</domain:period>
                <domain:authInfo>
                    <domain:pw>RMO7u1b2</domain:pw>
                </domain:authInfo>
            </domain:transfer>
        </transfer>
        <clTRID>DOMAIN_TRANSFER-qGPeMcZeJ9CA58Gq9huTA2</clTRID>
    </command>
</epp>
```

`domain:transfer` response with operation `request` example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <domain:trnData
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>trannsfer.id</domain:name>
                <domain:trStatus>pending</domain:trStatus>
                <domain:reID>reseller</domain:reID>
                <domain:reDate>2024-03-11T21:44:53.051803+07:00</domain:reDate>
                <domain:acID>mc1</domain:acID>
                <domain:acDate>2024-03-11T21:44:53.051803+07:00</domain:acDate>
                <domain:exDate>2028-07-28T00:00:00Z</domain:exDate>
            </domain:trnData>
        </resData>
        <trID>
            <clTRID>DOMAIN_TRANSFER-qGPeMcZeJ9CA58Gq9huTA2</clTRID>
            <svTRID>01962f9c-328f-73a9-a3c6-e14f804437f6</svTRID>
        </trID>
    </response>
</epp>
```

`domain:transfer` command with operation `query` example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
```

```
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <transfer op=\"query\">
            <domain:transfer
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>trannsfer.id</domain:name>
            </domain:transfer>
        </transfer>
        <clTRID>DOMAIN_TRANSFER-aDUadXb1Ag23Ytj85HLAnM</clTRID>
    </command>
</epp>
```

`domain:transfer` response with operation `query` example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <domain:trnData
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>trannsfer.id</domain:name>
                <domain:trStatus>pending</domain:trStatus>
                <domain:reID>mci</domain:reID>
                <domain:reDate>2024-03-11T14:44:53.052Z</domain:reDate>
                <domain:acID>mc1</domain:acID>
                <domain:acDate>2024-03-11T14:44:53.052Z</domain:acDate>
                <domain:exDate>2027-07-28T00:00:00Z</domain:exDate>
            </domain:trnData>
        </resData>
        <trID>
            <clTRID>DOMAIN_TRANSFER-aDUadXb1Ag23Ytj85HLAnM</clTRID>
            <svTRID>01962f9f-881b-700b-9505-8185b7aaa897</svTRID>
        </trID>
    </response>
</epp>
```

4.2.9 domain:update

The `domain:update` command is used to update domain name information. This command can also generate the `domain:authInfo` (transfer code), which is then sent to the domain holder for transferring the domain name.

If no changes are made during the `domain:update` request, the system will return error code **2308** (**"Data management policy violation"**).

If Media Cloud Indonesia contact IDs are still registered at the time of the update (usually after a transfer), the registrar must assign their contact IDs using `domain:update`. If only the holder's contact ID is updated, the technical contact field will be left empty, and any existing technical contact will be deleted.

The `domain:authInfo` (transfer code) must follow these rules:

- 6 to 12 characters (character sets according to Paragraph 3.2)
- No spaces, commas, or semicolons

`domain:update` command example with DNSSEC Extension

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <update>
            <domain:update
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>hello123456.id</domain:name>
                <domain:add>
                    <domain:ns>
                        <domain:hostAttr>
                            <domain:hostName>ns1.hello123456.id</domain:hostName>
                            <domain:hostAddr ip=\"v4\">1.0.30.0</domain:hostAddr>
                        </domain:hostAttr>
                    </domain:ns>
                </domain:add>
                <domain:rem>
                    <domain:ns>
                        <domain:hostAttr>
                            <domain:hostName>ns1.hello123456.id</domain:hostName>
                            <domain:hostAddr ip=\"v4\">1.0.20.0</domain:hostAddr>
                        </domain:hostAttr>
                    </domain:ns>
                </domain:rem>
            </domain:update>
        </update>
        <extension>
            <secDNS:update
                xmlns:secDNS=\"urn:ietf:params:xml:ns:secDNS-1.1\">
                <secDNS:add>
                    <secDNS:dsData>
                        <secDNS:keyTag>4985</secDNS:keyTag>
                        <secDNS:alg>3</secDNS:alg>
                        <secDNS:digestType>1</secDNS:digestType>
                        <secDNS:digest>48656c6c6f20576f726c64</secDNS:digest>
                    </secDNS:dsData>
                </secDNS:add>
                <secDNS:chg>
                    <secDNS:maxSigLife>4000</secDNS:maxSigLife>
                </secDNS:chg>
            </secDNS:update>
        </extension>
        <clTRID>DOMAIN_UPDATE-qdERJNkyu8jgX8EtFiH51x</clTRID>
    </command>
</epp>
```

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

```
<extension>
    <sedDNS:rem>
        <sedDNS:dsData>
            <sedDNS:keyTag>12345</sedDNS:keyTag>
            <sedDNS:alg>3</sedDNS:alg>
            <sedDNS:digestType>1</sedDNS:digestType>
            <sedDNS:digest>other-digest</sedDNS:digest>
        </sedDNS:dsData>
```

```
    </sedDNS:rem>
</extension>

```

`domain:update` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>DOMAIN_UPDATE-qdERJNkyu8jgX8EtFiH51x</clTRID>
            <svTRID>01962784-1a8d-776e-abf1-e040ca6a40b8</svTRID>
        </trID>
    </response>
</epp>
```

`domain:update` with Redemption Grace Period (RGP) command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <update>
            <domain:update
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>hello123456.id</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>
        <clTRID>DOMAIN_UPDATE-qdERJNkyu8jgX8EtFiH51x</clTRID>
    </command>
</epp>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<epp
    xmlns="urn:ietf:params:xml:ns:epp-1.0">
    <response>
        <result code="1000">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>DOMAIN_UPDATE-qdERJNkyu8jgX8EtFiH51x</clTRID>
            <svTRID>01962784-1a8d-776e-abf1-e040ca6a40b8</svTRID>
        </trID>

```

```
    </resp
```

4.2.10 domain:renew

Registrar-managed domain names are automatically renewed for one year upon expiration. However, domain names can also be renewed before they expire.

The `domain:renew` command is used to explicitly renew a domain for a specified period. When renewing a domain, the following information must be provided:

- The domain name
- The renewal period (e.g., `period = 5`)
- The current expiration date (e.g., `2000-04-03`)

`domain:renew` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <renew>
            <domain:renew
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>hello123456.id</domain:name>
                <domain:curExpDate>2025-04-11</domain:curExpDate>
                <domain:period unit=\"y\">1</domain:period>
            </domain:renew>
        </renew>
        <clTRID>DOMAIN_RENEW-h466DqUJVYfRJZboRuF9Gs</clTRID>
    </command>
</epp>\n
```

`domain:renew` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <domain:renData
                xmlns:domain=\"urn:ietf:params:xml:ns:domain-1.0\">
                <domain:name>hello123456.id</domain:name>
                <domain:exDate>2026-04-11T00:00:00Z</domain:exDate>
            </domain:renData>
        </resData>
        <trID>
            <clTRID>DOMAIN_RENEW-h466DqUJVYfRJZboRuF9Gs</clTRID>
            <svTRID>01962788-15c2-7217-8521-573185dbc21b</svTRID>
        </trID>
    </response>
</epp>
```

4.3 Contact commands

4.3.1 General information

In addition to the information below, contact commands are explained in detail in Reference (3). Only data allowed by law is published (see Paragraph 3.2).

4.3.2 contact:check

The `contact:check` command is used to verify whether a contact with certain id exists.

`contact:check` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <check
            xmlns:contact=\"urn:ietf:params:xml:ns:contact-1.0\">
            <contact:check>
                <contact:id>nIuh4WezutFelfaz</contact:id>
            </contact:check>
        </check>
        <clTRID>CONTACT_CHECK-wMy9UgGUHDiwtYSKRE1ZNv</clTRID>
    </command>
</epp>
```

`contact:check` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <contact:chkData
                xmlns:contact=\"urn:ietf:params:xml:ns:contact-1.0\">
                <contact:cd>
                    <contact:id avail=\"1\">nIuh4WezutFelfaz</contact:id>
                </contact:cd>
            </contact:chkData>
        </resData>
        <trID>
            <clTRID>CONTACT_CHECK-mci-wMy9UgGUHDiwtYSKRE1ZNv</clTRID>
            <svTRID>01962541-b68b-7f26-9b68-0c3ae8d3bc41</svTRID>
        </trID>
    </response>
</epp>
```

4.3.3 contact:info

The `contact:info` command is used to request contact information. The `<contact:email>` field is only disclosed for contacts managed by the registrar submitting the query.

`contact:info` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
```

```
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <info
            xmlns:contact=\"urn:ietf:params:xml:ns:contact-1.0\">
            <contact:info>
                <contact:id>K8xqlqX3DaYKs7vX</contact:id>
            </contact:info>
        </info>
        <clTRID>CONTACT_INFO-pXa34bbMAQQkvTvpcvsQbX</clTRID>
    </command>
</epp>
```

`contact:info` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <contact:infData
                xmlns:contact=\"urn:ietf:params:xml:ns:contact-1.0\">
                <contact:id>K8xqlqX3DaYKs7vX</contact:id>
                <contact:status s=\"ok\" lang=\"en\"></contact:status>
                <contact:postalInfo type=\"loc\">
                    <contact:name>Hello</contact:name>
                    <contact:org>Organization</contact:org>
                    <contact:addr>
                        <contact:street>street 1</contact:street>
                        <contact:city>Bekasi</contact:city>
                        <contact:sp>Jakarta</contact:sp>
                        <contact:pc>17145</contact:pc>
                        <contact:cc>ID</contact:cc>
                    </contact:addr>
                </contact:postalInfo>
                <contact:voice>+62.8123456790</contact:voice>
                <contact:fax>+62.8123456790</contact:fax>
                <contact:email>hello@gmail.com</contact:email>
                <contact:clID>nntwIWVG14EVnuT0</contact:clID>
                <contact:crID>nntwIWVG14EVnuT0</contact:crID>
                <contact:crDate>2025-04-11T20:37:49.449+07:00</contact:crDate>
                <contact:upDate>2025-04-11T20:37:49.449+07:00</contact:upDate>
            </contact:infData>
        </resData>
        <trID>
            <clTRID>CONTACT_INFO-pXa34bbMAQQkvTvpcvsQbX</clTRID>
            <svTRID>01962515-4e53-77bc-9582-28ef5a78e5b2</svTRID>
        </trID>
    </response>
</epp>
```

4.3.4 contact:create

The `contact:create` command is used to generate contact IDs. Each reseller must create and update their contact IDs for the domain names they manage. This is especially important during transfers, as no updates can be made to a domain name until the registrar assigns their contact IDs.

The following rules apply:

- `<contact:postalInfo type="loc">`: Only the **"loc"** (localized) address type is supported. The **"int"** (international) address type using 7-bit ASCII characters is not allowed and will be rejected. If both **"int"** and **"loc"** types are submitted, the **"int"** type will be ignored.
- `contact:street`: The address can contain up to three lines but must have at least one `contact:street` element. If there is a PO Box, it should be entered in this field.
- `contact:disclose`: This element is not supported.
- `contact:email`: The holder's email address must be provided. The reseller's email address must not be entered.
- `contact:pw`: This field is not used by Media Cloud Indonesia and is not stored. Authentication is performed during the EPP login process.
- `contact:id`: Must be unique, contain at least one uppercase ASCII letter, and may include numbers and hyphens.

`contact:create` command example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp
    xmlns="urn:ietf:params:xml:ns:epp-1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
epp-1.0.xsd">
    <command>
        <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>PhOabzuRunKgFivD</contact:id>
                <contact:postalInfo type=\"int\">
                    <contact:name>Hello</contact:name>
                    <contact:org>Organization</contact:org>
                    <contact:addr>
                        <contact:street>street 1</contact:street>
                        <contact:city>Bekasi</contact:city>
                        <contact:sp>Jakarta</contact:sp>
                        <contact:pc>17145</contact:pc>
                        <contact:cc>ID</contact:cc>
                    </contact:addr>
                </contact:postalInfo>
                <contact:voice>+62.8123456790</contact:voice>
                <contact:fax>+62.8123456790</contact:fax>
                <contact:email>hello@gmail.com</contact:email>
                <contact:authInfo>
                    <contact:pw/>
                </contact:authInfo>
            </contact:create>
        </create>
        <clTRID>CONTACT_CREATE-7KiqxpDJV1ujcGdVwgPX9R</clTRID>
    </command>
</epp>
```

`contact:create` response example

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<epp
    xmlns="urn:ietf:params:xml:ns:epp-1.0">
    <response>
        <result code="1000">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <contact:creData
                xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
                <contact:id>Do5o6kn6Ze36ZneE</contact:id>
                <contact:crDate>2024-05-11T20:10:01.036422+07:00</contact:crDate>
            </contact:creData>
        </resData>
        <trID>
            <clTRID>CONTACT_CREATE-kFk9BcNaNBByZduEMRodgt</clTRID>
            <svTRID>019624f8-b3c0-7be6-95fc-d72655143f0a</svTRID>
        </trID>
    </response>
</epp>
```

4.3.5 contact:delete

The `contact:delete` command can be used to delete contact IDs, provided they are not linked to any domain name or object (`linked` status).

Only contact IDs managed by the registrar can be deleted. Registrars are required to delete any of their contact IDs that are no longer in use.

`contact:delete` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <delete xmlns:contact=\"urn:ietf:params:xml:ns:contact-1.0\">
            <contact:delete>
                <contact:id>K8xqlqX3DaYKs7vX</contact:id>
            </contact:delete>
        </delete>
        <clTRID>CONTACT_DELETE-quoXLabGRfi6qptMTpffYo</clTRID>
    </command>
</epp>\n
```

`contact:delete` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>CONTACT_DELETE-quoXLabGRfi6qptMTpffYo</clTRID>
            <svTRID>01962537-c4cd-739d-b9d9-4b3408bb9870</svTRID>
        </trID>
    </response>
</epp>
```

4.3.6 contact:transfer

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

4.3.7 contact:update

The `contact:update` command is used to update contact IDs. The same conditions as **a** `contact:create` apply. The `contact:addr` element can only be updated as a whole block.

The following rules apply:

- `<contact:postalInfo type="loc">`: Only the `loc` (localized) address type is supported. The `int` (international) address type with 7-bit ASCII characters is not allowed and will be rejected. If both `int` and `loc` types are submitted, the `int` type will be ignored.
- `contact:street`: The address can contain up to three lines but must have at least one `contact:street` element. If there is a PO Box, it should be entered in this field.
- `contact:disclose`: This element is not supported.
- `contact:email`: The holder's email address must be specified. The registrar's email address must not be entered in this field.
- `contact:pw`: This field is not used by Media Cloud and is not stored. Authentication is performed during the EPP login process.
- `contact:id`: Must be unique on the EPP server, contain at least one uppercase ASCII letter, and may include numbers and hyphens.
- `contact:update` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <update>
            <contact:update
                xmlns:contact=\"urn:ietf:params:xml:ns:contact-1.0\">
                <contact:id>K8xqlqX3DaYKs7vX</contact:id>
                <contact:chg>
                    <contact:postalInfo type=\"int\">
                        <contact:name>URIERS</contact:name>
                        <contact:org>officia cillum</contact:org>
                        <contact:addr>
                            <contact:street>490 Conroy Meadows</contact:street>
                            <contact:street>1359 Sauer Fall</contact:street>
                            <contact:city>Bekasi</contact:city>
                            <contact:pc>17433</contact:pc>
                            <contact:cc>ID</contact:cc>
                        </contact:addr>
                    </contact:postalInfo>
                    <contact:voice>+62.8123456789</contact:voice>
                    <contact:fax>+62.8123456789</contact:fax>
                    <contact:email>lexus@mail.id</contact:email>
                    <contact:authInfo>
                        <contact:pw/>
                    </contact:authInfo>
                </contact:chg>
            </contact:update>
        </update>
        <clTRID>CONTACT_UPDATE-sZtdeSyifBoWH19Kf2HLHm</clTRID>
    </command>
</epp>
```

`contact:update` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>CONTACT_UPDATE-sZtdeSyifBoWH19Kf2HLHm</clTRID>
            <svTRID>01962527-8431-72da-a9b0-bb8cc5738a08</svTRID>
        </trID>
    </response>
</epp>
```

4.4 Host commands

4.4.1 General information

In addition to the information below, all **host** commands are explained in detail in Reference (4).

There are three types of name servers (hosts):

Type	Description	Example
Valid internal name server under .id	Ends with **.id** and the parent domain name is registered	**ns1.iamregistered.id**
Invalid internal name server under .id	Ends with **.id** but the parent domain name is not registered	**ns1.iamnotregistered.id**
External name server not under .id	Does not end with **.id**	**ns1.yourname.com**

Valid internal name servers under **.id** are linked to the domain name holder and managed by their registrar. They are automatically transferred together with the domain name.

External name servers (e.g., **ns1.yourname.com**) cannot be registered in the registry.

4.4.2 host:check

The `host:check` command is used to verify whether name servers exist. The server will respond with **registered** or **not registered**.

- A maximum of **10 hosts** can be queried at once using `host:check`.
- If more than 10 hosts are queried at the same time, the server will return error code **2308** (**“Data management policy violation”**).

`host:check` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <check
            xmlns:host=\"urn:ietf:params:xml:ns:host-1.0\">
            <host:check>
                <host:name>ns1.hello123456.id</host:name>
                <host:name>ns2.hello123456.id</host:name>
```

```
            </host:check>
        </check>
        <clTRID>HOST_CHECK-ckS6pLz6XzghNm9uqeVqL5</clTRID>
    </command>
</epp>
```

`host:check` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <host:chkData
                xmlns:host=\"urn:ietf:params:xml:ns:host-1.0\">
                <host:cd>
                    <host:name avail=\"0\">ns1.hello123456.id</host:name>
                    <host:reason>In Use</host:reason>
                </host:cd>
                <host:cd>
                    <host:name avail=\"1\">ns2.hello123456.id</host:name>
                </host:cd>
            </host:chkData>
        </resData>
        <trID>
            <clTRID>HOST_CHECK-ckS6pLz6XzghNm9uqeVqL5</clTRID>
            <svTRID>01962582-56fe-7b8d-af0b-f3a837eedcb7</svTRID>
        </trID>
    </response>
</epp>
```

4.4.3 host:info

The `host:info` command is used to request detailed information about name servers. The following information will only be disclosed if the name servers are administered by the requesting registrar:

- `<host:clID>` : ID of the sponsoring client (current registrar).
- `<host:crID>` : ID of the registrar who created the name server.
- `<host:crDate>` : Date when the name server was created.

`host:info` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <info
            xmlns:host=\"urn:ietf:params:xml:ns:host-1.0\">
            <host:info>
                <host:name>ns1.hello123456.id</host:name>
            </host:info>
        </info>
        <clTRID>HOST_INFO-pDmUugFxoZF1tyhTh5mX5a</clTRID>
    </command>
```

```
</epp>
```

`host:info` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <host:infData
                xmlns:host=\"urn:ietf:params:xml:ns:host-1.0\">
                <host:name>ns1.hello123456.id</host:name>
                <host:roid>MCI_Tf4wLCHGNxquvviQiPAzVx-11577869</host:roid>
                <host:status s=\"ok\"></host:status>
                <host:addr ip=\"v4\">172.30.13.21</host:addr>
                <host:addr ip=\"v4\">172.30.13.21</host:addr>
                <host:clID>nntwIWVG14EVnuT0</host:clID>
                <host:crID>nntwIWVG14EVnuT0</host:crID>
                <host:crDate>2025-04-11T22:30:56.192+07:00</host:crDate>
                <host:upDate>2025-04-11T22:34:39.859+07:00</host:upDate>
            </host:infData>
        </resData>
        <trID>
            <clTRID>HOST_INFO-pDmUugFxoZF1tyhTh5mX5a</clTRID>
            <svTRID>0196257f-e99b-7ab6-b8e5-f7b77a06cb0e</svTRID>
        </trID>
    </response>
</epp>
```

4.4.4 host:create

The `host:create` command is used to register name servers. The authorizations listed in **Paragraph 4.4.1** apply.

- **External and invalid internal name servers** can be created by anyone.
- **Valid internal name servers** can only be created by the registrar who manages the parent domain name.

The following limits apply:

- A maximum of **20 IP addresses** per host.
- A maximum of **20 hosts** per domain name.

`host:create` command example

```
?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <create>
            <host:create
                xmlns:host=\"urn:ietf:params:xml:ns:host-1.0\">
                <host:name>ns1.hello123456.id</host:name>
                <host:addr ip=\"v4\">172.30.13.21</host:addr>
            </host:create>
        </create>
        <clTRID>HOST_CREATE-ghLawuPrYSSxSAHrZJ8zGy</clTRID>
```

```
    </command>
</epp>\n
```

`host:create` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <resData>
            <host:creData
                xmlns:host=\"urn:ietf:params:xml:ns:host-1.0\">
                <host:name>ns1.hello123456.id</host:name>
                <host:crDate>2024-03-10T00:00:00Z</host:crDate>
            </host:creData>
        </resData>
        <trID>
            <clTRID>HOST_CREATE-ghLawuPrYSSxSAHrZJ8zGy</clTRID>
            <svTRID>01962579-aedc-70ee-8f81-3c30cb10aae5</svTRID>
        </trID>
    </response>
</epp>
```

4.4.5 host: transfer

This command is **not supported** as stated in **Reference (03)**, and any request will be **rejected with error code 2000**.

Valid internal name servers are associated with the holder of the corresponding domain name, and they will be **automatically transferred along with the domain name**.

4.4.6 host:delete

The `host:delete` command is used to **delete name servers**.

Name servers can only be deleted if they are **no longer referenced** in the registration system, meaning **no domain name is pointing to the name server**.

`host:delete` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <delete
            xmlns:host=\"urn:ietf:params:xml:ns:host-1.0\">
            <host:delete>
                <host:name>ns1.hello123456.id</host:name>
            </host:delete>
        </delete>
        <clTRID>HOST_DELETE-pFvLUXNAWRQ8sscaCgvRqQ</clTRID>
    </command>
</epp>
```

`host:delete` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
```

```
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>HOST_DELETE-mci-pFvLUXNAWRQ8sscaCgvRqQ</clTRID>
            <svTRID>01962584-3b17-798d-babf-c103b29ab14a</svTRID>
        </trID>
    </response>
</epp>
```

4.4.7 host:update

update the `host:update` command is used to update **name servers**.

`host:update` command example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\"
    xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"urn:ietf:params:xml:ns:epp-1.0 epp-1.0.xsd\">
    <command>
        <update
            xmlns:host=\"urn:ietf:params:xml:ns:host-1.0\">
            <host:update>
                <host:name>ns1.hello123456.id</host:name>
                <host:add>
                    <host:addr ip=\"v4\">172.30.13.21</host:addr>
                </host:add>
                <host:rem>
                    <host:addr ip=\"v4\">0.1.0.1</host:addr>
                </host:rem>
            </host:update>
        </update>
        <clTRID>HOST_UPDATE-2GKYsvBqQoszsM9WHgWAxr</clTRID>
    </command>
</epp>\n
```

`host:update` response example

```
<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
<epp
    xmlns=\"urn:ietf:params:xml:ns:epp-1.0\">
    <response>
        <result code=\"1000\">
            <msg>Command completed successfully</msg>
        </result>
        <trID>
            <clTRID>HOST_UPDATE-2GKYsvBqQoszsM9WHgWAxr</clTRID>
            <svTRID>0196257d-194c-7991-8bd2-37d850bd7606</svTRID>
        </trID>
    </response>
</epp>
```

Appendix A: Abbreviations and Terms

Term	Explanation
AuthInfo	The transfer code that is required to transfer a domain name.
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 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 Media Cloud Indonesia, this is a name server that does not end with .id.
Internal name server	A name server that belongs to the ccTLD administered by the registration system. At Media Cloud Indonesia, this is a name server that ends with .id.
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 Media Cloud Indonesia
Registrant	End customer who owns a domain name.
Registry	Organization 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.