
Workgroup:	Network Working Group
Internet-Draft:	draft-wullink-restful-epp-json-00
Published:	1 February 2024
Intended Status:	Standards Track
Expires:	4 August 2024
Authors:	M. Wullink M. Davids
	<i>SIDN Labs SIDN Labs</i>

XML to JSON Conversion rules for EPP

Abstract

This document describes the rules for converting The Extensible Provisioning Protocol (EPP) [RFC5730] XML based messages and the corresponding XSD schemas to a JSON [RFC8259] and JSON schema [REF-TO-JSON-SCHEMA-HERE] for use with EPP and RESTful EPP[REF-TO-REPP-HERE].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 4 August 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

- 1. Introduction
 - 1.1. Motivation
- 2. Terminology
- 3. Conventions Used in This Document
- 4. Conversion Rules
 - 4.1. Empty
 - 4.2. Pure text content
 - 4.3. Attributes only
 - 4.4. Pure text content and attributes
 - 4.5. Child elements with different names
 - 4.6. Child elements with identical names
 - 4.7. Child elements and contiguous text
- 5. Examples
 - 5.1. Hello
 - 5.2. Login
 - 5.3. Logout
 - 5.4. Check
 - 5.5. Info
 - 5.6. Poll
 - 5.7. Poll Ack
 - 5.8. Transfer Query
 - 5.9. Create
 - 5.10. Delete
 - 5.11. Renew
 - 5.12. Transfer Request
 - 5.13. Transfer Cancel
 - 5.14. Transfer Reject
 - 5.15. Transfer Approve

[5.16. Update](#)[6. IANA Considerations](#)[7. Internationalization Considerations](#)[8. Security Considerations](#)[9. Acknowledgments](#)[10. Normative References](#)[11. Informative References](#)[Appendix A. Appendix A. Media Type Registration: application/epp+json](#)[Authors' Addresses](#)

1. Introduction

This document describes rules for converting valid EPP XML messages to the JavaScript Object Notation (JSON) Data Interchange Format [RFC8259], for use with EPP, RESTful EPP[REF-TO-REPP-HERE] and future EPP related protocols and transports. The Extensible Provisioning Protocol (EPP) [RFC5730] describes an XML based protocol, which is defined by XML Schema Definition (XSD). The XSDs are published as part of the EPP RFCs and contain the formal syntax for EPP XML message and provide a method for validation of XML messages. JSON Schema [REF-TO-JSON-SCHEMA-HERE] is a method of specifying formal syntax for JSON format.

1.1. Motivation

RESTful EPP introduces a new transport mechanism for EPP messages that aligns more closely with modern cloud infrastructure, enhancing the scalability of EPP server deployments. While RESTful protocols do not mandate a specific media type for resource description, the widespread adoption of JSON in web services has established it as the de facto standard for modern APIs. The increasing availability of tools, software libraries, and a skilled workforce, coupled with the declining popularity of XML, has led several registries to adopt JSON for data exchange within their API ecosystems. By extending EPP to support JSON, registries can offer a unified API ecosystem that extends beyond domain name and IP address provisioning, maintaining a consistent technology stack, data formats, and developer experience.

JSON's syntax, known for its straightforwardness and minimal verbosity compared to XML, significantly eases the tasks of writing, reading, and maintaining code. This simplicity is especially advantageous for the rapid comprehension and integration of provisioning APIs.

The lightweight nature of JSON can result in faster processing and data transfers, a critical aspect in high-volume transaction environments such as domain registration. Enhanced API response times can lead to more efficient domain lookups, registrations, and updates. Moreover, JSON parsing is typically faster and more straightforward than XML parsing, contributing to improved system performance amid frequent interactions between EPP clients and servers.

However, the absence of a standardized JSON format for domain provisioning has led to the emergence of TLD-specific implementations that lack interoperability, increasing the development effort required for integration. Similarly, at the registrar level, the absence of standards has resulted in numerous incompatible API implementations provided to clients and resellers. Standardizing a JSON format for domain provisioning within the EPP framework could mitigate these challenges, reducing fragmentation and simplifying integration efforts across the domain registration industry.

2. Terminology

In this document the following terminology is used.

EPP RFCs - This is a reference to the EPP version 1.0 specifications [[RFC5730](#)], [[RFC5731](#)], [[RFC5732](#)] and [[RFC5733](#)].

RESTful EPP or REPP - The RESTful transport for EPP described in this document.

3. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

JSON is case sensitive. Unless stated otherwise, JSON specifications and examples provided in this document MUST be interpreted in the character case presented. The examples in this document assume that request and response messages are properly formatted JSON documents. Indentation and white space in examples are provided only to illustrate element relationships and for improving readability, and are not REQUIRED features of the protocol.

4. Conversion Rules

A XML element may exist in one of 7 distinct forms, the sections below describe how these forms MUST be translated to valid JSON.

1. Empty
2. Pure text content
3. Attributes only
4. Pure text content and attributes
5. Child elements with different names

6. Child elements with identical names
7. Child element(s) and contiguous text

4.1. Empty

An empty XML element **MUST** be mapped to a key matching the name of the element and a null value.

XML:

```
<hello/>
```

JSON:

```
{
  "hello": null
}
```

4.2. Pure text content

An XML element containing text only **MUST** be mapped to a key matching the name of the element and the text **MUST** be used for the value

XML:

```
<lang>en</lang>
```

JSON:

```
{
  "lang": "en"
}
```

4.3. Attributes only

An XML element containing one or more attributes only, **MUST** be mapped to a JSON object matching the name of the element. Each XML attribute, prefixed using the @ character, **MUST** be added as a key-value pair to the object.

XML:

```
<msgQ count="5" id="12345"/>
```

JSON:

```
{
  "msgQ": {
    "@count": "5",
    "@id": "12345"
  }
}
```

4.4. Pure text content and attributes

An XML element containing one or more attributes and text content only, MUST be mapped to a JSON object matching the name of the element. The text content MUST, prefixed using the string `#text`, MUST be added as a key-value pair to the object.

XML:

```
<msg lang="en">Command completed successfully</msg>
```

JSON:

```
{
  "msg": {
    "@lang": "en",
    "#text": "Command completed successfully"
  }
}
```

4.5. Child elements with different names

An XML element containing one or more child elements, where each child uses an unique name, MUST be mapped to a JSON object matching the name of the element. Each child element MUST be added as a key-value pair to the parent object.

XML:

```
<trID>
  <clTRID>ABC-12345</clTRID>
  <svTRID>54321-XYZ</svTRID>
</trID>
```

JSON:

```
{
  "trID": {
    "clTRID": "ABC-12345",
    "svTRID": "54321-XYZ"
  }
}
```

4.6. Child elements with identical names

An XML element containing multiple child elements, where multiple child elements use the same name, **MUST** be mapped to a JSON object containing an array. The name of the array **MUST** match the name of the non-unique children, each child element **MUST** be converted to JSON and added to the array.

XML:

```
<host>
  <addr>192.0.2.1</addr>
  <addr>192.0.2.2</addr>
</host>
```

JSON:

```
{
  "host": {
    "addr": [
      "192.0.2.1",
      "192.0.2.2"
    ]
  }
}
```

4.7. Child elements and contiguous text

An XML element containing one or more child elements and contiguous text, **MUST** be mapped to a JSON object containing a key-value entry for each child element, the text value **MUST** result in a key named `#text`.

XML:

```
<msg lang="en">
  Credit balance low.
  <limit>100</limit>
  <bal>5</bal>
</msg>
```

JSON:

```
{
  "msg": {
    "@lang": "en",
    "limit": 100,
    "bal": 5,
    "#text": "Credit balance low."
  }
}
```

When child elements are mixed with multiple text segments, the resulting `#text` key-value entry MUST be an array, containing all text segments.

XML:

```
<msg lang="en">
  Credit balance low.
  <limit>100</limit>
  <bal>5</bal>
  Please increase balance.
</msg>
```

JSON:

```
{
  "msg": {
    "@lang": "en",
    "limit": 100,
    "bal": 5,
    "#text": ["Credit balance low.", "Please increase balance asap."]
  }
}
```

The rules above are based on the conversion approach found on [\[XMLCOM-WEB\]](#)

5. Examples

This section lists examples for each of the existing EPP commands that are support by REPP.

5.1. Hello

The Hello request message does not exist in the context of REPP.

Example XML response:


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp
  xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <greeting>
    <svID>Example EPP server epp.example.com</svID>
    <svDate>2000-06-08T22:00:00.0Z</svDate>
    <svcMenu>
      <version>1.0</version>
      <lang>en</lang>
      <lang>fr</lang>
      <objURI>urn:ietf:params:xml:ns:obj1</objURI>
      <objURI>urn:ietf:params:xml:ns:obj2</objURI>
      <objURI>urn:ietf:params:xml:ns:obj3</objURI>
      <svcExtension>
        <extURI>http://custom/obj1ext-1.0</extURI>
      </svcExtension>
    </svcMenu>
    <dcp>
      <access>
        <all/>
      </access>
      <statement>
        <purpose>
          <admin/>
          <prov/>
        </purpose>
        <recipient>
          <ours/>
          <public/>
        </recipient>
        <retention>
          <stated/>
        </retention>
      </statement>
    </dcp>
  </greeting>
</epp>
```

Example JSON response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "greeting": {
      "svID": "Example REPP server v1.0",
      "svDate": "2000-06-08T22:00:00.0Z",
      "svcMenu": {
        "version": "1.0",
        "lang": [
          "en",
          "fr"
        ]
      },
      "dcp": {
        "access": {
          "all": null
        },
        "statement": {
          "purpose": {
            "admin": null,
            "prov": null
          },
          "recipient": {
            "ours": null,
            "public": null
          },
          "retention": {
            "stated": null
          }
        }
      }
    }
  }
}
```

5.2. Login

The Login request and response message are not used for REPP.

5.3. Logout

The Logout request and response message are not used for REPP.

5.4. Check

The Check request and responses messages are not used for REPP.

5.5. Info

The Info request message is not used for REPP.

Example XML Domain Info 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>
    <resData>
      <domain:infData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:roid>EXAMPLE1-REP</domain:roid>
        <domain:status s="ok"/>
        <domain:registrar>jd1234</domain:registrar>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:ns>
          <domain:hostObj>ns1.example.com</domain:hostObj>
          <domain:hostObj>ns1.example.net</domain:hostObj>
        </domain:ns>
        <domain:host>ns1.example.com</domain:host>
        <domain:host>ns2.example.com</domain:host>
        <domain:clID>ClientX</domain:clID>
        <domain:crID>ClientY</domain:crID>
        <domain:crDate>1999-04-03T22:00:00.0Z</domain:crDate>
        <domain:upID>ClientX</domain:upID>
        <domain:upDate>1999-12-03T09:00:00.0Z</domain:upDate>
        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
        <domain:trDate>2000-04-08T09:00:00.0Z</domain:trDate>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:infData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54322-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Example JSON Domain Info response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1000",
        "msg": "Command completed successfully"
      },
      "resData": {
        "domain:infData": {
          "@xmlns:domain": "urn:ietf:params:xml:ns:domain-1.0",
          "domain:name": "example.com",
          "domain:roid": "EXAMPLE1-REP",
          "domain:status": {
            "@s": "ok"
          },
          "domain:registrant": "jd1234",
          "domain:contact": [
            {
              "@type": "admin",
              "#text": "sh8013"
            },
            {
              "@type": "tech",
              "#text": "sh8013"
            }
          ],
          "domain:ns": {
            "domain:hostObj": [
              "ns1.example.com",
              "ns1.example.net"
            ]
          },
          "domain:host": [
            "ns1.example.com",
            "ns2.example.com"
          ],
          "domain:clID": "ClientX",
          "domain:crID": "ClientY",
          "domain:crDate": "1999-04-03T22:00:00.0Z",
          "domain:upID": "ClientX",
          "domain:upDate": "1999-12-03T09:00:00.0Z",
          "domain:exDate": "2005-04-03T22:00:00.0Z",
          "domain:trDate": "2000-04-08T09:00:00.0Z",
          "domain:authInfo": {
            "domain:pw": "2fooBAR"
          }
        },
        "trID": {
          "clTRID": "ABC-12345",
          "svTRID": "54322-XYZ"
        }
      }
    }
  }
}
```

5.6. Poll

The Poll request message is not used for REPP.

Example XML response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp
  xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1301">
      <msg>Command completed successfully; ack to dequeue</msg>
    </result>
    <msgQ count="4" id="12346">
      <qDate>2000-06-08T22:10:00.0Z</qDate>
      <msg lang="en">Credit balance low.
        <limit>100</limit>
        <bal>5</bal>
      </msg>
    </msgQ>
    <trID>
      <clTRID>ABC-12346</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Example JSON response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1301",
        "msg": "Command completed successfully; ack to dequeue"
      },
      "msgQ": {
        "@count": "4",
        "@id": "12346",
        "qDate": "2024-01-15T22:10:00.0Z",
        "msg": {
          "@lang": "en",
          "limit": "100",
          "bal": "5",
          "#text": "Credit balance low."
        }
      },
      "trID": {
        "clTRID": "ABC-12346",
        "svTRID": "54321-XYZ"
      }
    }
  }
}
```

5.7. Poll Ack

The Poll Ack request message is not used for REPP.

Example XML 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>
    <msgQ count="0" id="12345"/>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>XYZ-12345</svTRID>
    </trID>
  </response>
</epp>
```

Example JSON response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1000",
        "msg": "Command completed successfully"
      },
      "msgQ": {
        "@count": "0",
        "@id": "12345"
      },
      "trID": {
        "clTRID": "ABC-12345",
        "svTRID": "XYZ-12345"
      }
    }
  }
}
```

5.8. Transfer Query

The Domain Transfer Query request message is not used for REPP.

Example XML Domain Transfer Query 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>
    <resData>
      <domain:trnData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:trStatus>pending</domain:trStatus>
        <domain:reID>ClientX</domain:reID>
        <domain:reDate>2000-06-06T22:00:00.0Z</domain:reDate>
        <domain:acID>ClientY</domain:acID>
        <domain:acDate>2000-06-11T22:00:00.0Z</domain:acDate>
        <domain:exDate>2002-09-08T22:00:00.0Z</domain:exDate>
      </domain:trnData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54322-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Example JSON Domain Transfer Query response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1000",
        "msg": "Command completed successfully"
      },
      "resData": {
        "domain:trnData": {
          "@xmlns:domain": "urn:ietf:params:xml:ns:domain-1.0",
          "domain:name": "example.com",
          "domain:trStatus": "pending",
          "domain:reID": "ClientX",
          "domain:reDate": "2000-06-06T22:00:00.0Z",
          "domain:acID": "ClientY",
          "domain:acDate": "2000-06-11T22:00:00.0Z",
          "domain:exDate": "2002-09-08T22:00:00.0Z"
        },
        "trID": {
          "clTRID": "ABC-12345",
          "svTRID": "54322-XYZ"
        }
      }
    }
  }
}
```

5.9. Create

Example XML Domain Create request:


```
<?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>example.com</domain:name>
        <domain:period unit="y">2</domain:period>
        <domain:ns>
          <domain:hostObj>ns1.example.net</domain:hostObj>
          <domain:hostObj>ns2.example.net</domain:hostObj>
        </domain:ns>
        <domain:registrant>jd1234</domain:registrant>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:create>
    </create>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example JSON Domain Create request:

```
{
  "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": "example.com",
          "domain:period": {
            "@unit": "y",
            "#text": "2"
          },
        },
        "domain:ns": {
          "domain:hostObj": [
            "ns1.example.net",
            "ns2.example.net"
          ],
        },
        "domain:registrar": "jd1234",
        "domain:contact": [
          {
            "@type": "admin",
            "#text": "sh8013"
          },
          {
            "@type": "tech",
            "#text": "sh8013"
          }
        ],
        "domain:authInfo": {
          "domain:pw": "2fooBAR"
        }
      },
      "clTRID": "ABC-12345"
    }
  }
}
```

Example XML Domain Create 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>
    <resData>
      <domain:creData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</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>
```

Example JSON Domain Create response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1000",
        "msg": "Command completed successfully"
      },
      "resData": {
        "domain:creData": {
          "@xmlns:domain": "urn:ietf:params:xml:ns:domain-1.0",
          "domain:name": "example.com",
          "domain:crDate": "1999-04-03T22:00:00.0Z",
          "domain:exDate": "2001-04-03T22:00:00.0Z"
        }
      },
      "trID": {
        "clTRID": "ABC-12345",
        "svTRID": "54321-XYZ"
      }
    }
  }
}
```

5.10. Delete

The Delete request message is not used for REPP.

Example XML Domain Delete 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>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Example JSON Domain Delete response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1000",
        "msg": "Command completed successfully"
      },
      "trID": {
        "clTRID": "ABC-12345",
        "svTRID": "54321-XYZ"
      }
    }
  }
}
```

5.11. Renew

The Renew request message is not used for REPP.

Example XML Domain Renew 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>
    <resData>
      <domain:renData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:exDate>2005-04-03T22:00:00.0Z</domain:exDate>
      </domain:renData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54322-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Example JSON Domain Renew response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1000",
        "msg": "Command completed successfully"
      },
      "resData": {
        "domain:renData": {
          "@xmlns:domain": "urn:ietf:params:xml:ns:domain-1.0",
          "domain:name": "example.com",
          "domain:exDate": "2005-04-03T22:00:00.0Z"
        }
      },
      "trID": {
        "clTRID": "ABC-12345",
        "svTRID": "54322-XYZ"
      }
    }
  }
}
```

5.12. Transfer Request

The Transfer request message is not used for REPP.

Example XML Domain Transfer response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp
  xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1001">
      <msg>Command completed successfully; action pending</msg>
    </result>
    <resData>
      <domain:trnData
        xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>example.com</domain:name>
        <domain:trStatus>pending</domain:trStatus>
        <domain:reID>ClientX</domain:reID>
        <domain:reDate>2000-06-08T22:00:00.0Z</domain:reDate>
        <domain:acID>ClientY</domain:acID>
        <domain:acDate>2000-06-13T22:00:00.0Z</domain:acDate>
        <domain:exDate>2002-09-08T22:00:00.0Z</domain:exDate>
      </domain:trnData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54322-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Example JSON Domain Transfer response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1001",
        "msg": "Command completed successfully; action pending"
      },
      "resData": {
        "domain:trnData": {
          "@xmlns:domain": "urn:ietf:params:xml:ns:domain-1.0",
          "domain:name": "example.com",
          "domain:trStatus": "pending",
          "domain:reID": "ClientX",
          "domain:reDate": "2000-06-08T22:00:00.0Z",
          "domain:acID": "ClientY",
          "domain:acDate": "2000-06-13T22:00:00.0Z",
          "domain:exDate": "2002-09-08T22:00:00.0Z"
        },
        "trID": {
          "clTRID": "ABC-12345",
          "svTRID": "54322-XYZ"
        }
      }
    }
  }
}
```

5.13. Transfer Cancel

The Transfer Cancel request message is not used for REPP.

Example XML Domain Cancel Transfer 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>XYZ-12345</svTRID>
    </trID>
  </response>
</epp>
```

Example JSON Domain Cancel Transfer response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1000",
        "msg": "Command completed successfully"
      },
      "trID": {
        "clTRID": "ABC-12345",
        "svTRID": "XYZ-12345"
      }
    }
  }
}
```

5.14. Transfer Reject

The Transfer Reject request message is not used for REPP and the response message is the same as for the Transfer Cancel command.

5.15. Transfer Approve

The Transfer Approve request message is not used for REPP and the response message is the same as for the Transfer Cancel command.

5.16. Update

Example XML Domain Update request:

```
<?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>example.com</domain:name>
        <domain:chg>
          <domain:registrant>sh8013</domain:registrant>
          <domain:authInfo>
            <domain:pw>2BARfoo</domain:pw>
          </domain:authInfo>
        </domain:chg>
      </domain:update>
    </update>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Example JSON Domain Update request:

```
{
  "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": "example.com",
          "domain:chg": {
            "domain:registrant": "sh8013",
            "domain:authInfo": {
              "domain:pw": "2BARfoo"
            }
          }
        }
      },
      "clTRID": "ABC-12345"
    }
  }
}
```

Example XML Domain 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>XYZ-12345</svTRID>
    </trID>
  </response>
</epp>
```

Example JSON Domain Update response:

```
{
  "epp": {
    "@xmlns": "urn:ietf:params:xml:ns:epp-1.0",
    "response": {
      "result": {
        "@code": "1000",
        "msg": "Command completed successfully"
      },
      "trID": {
        "clTRID": "ABC-12345",
        "svTRID": "XYZ-12345"
      }
    }
  }
}
```

6. IANA Considerations

The new application/epp+json MIME media type is used in this document, the registration template is included in Appendix A.

7. Internationalization Considerations

TODO

8. Security Considerations

TODO

9. Acknowledgments

TODO

10. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <<https://www.rfc-editor.org/info/rfc5730>>.
- [RFC5731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", STD 69, RFC 5731, DOI 10.17487/RFC5731, August 2009, <<https://www.rfc-editor.org/info/rfc5731>>.
- [RFC5732] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Host Mapping", STD 69, RFC 5732, DOI 10.17487/RFC5732, August 2009, <<https://www.rfc-editor.org/info/rfc5732>>.
- [RFC5733] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Contact Mapping", STD 69, RFC 5733, DOI 10.17487/RFC5733, August 2009, <<https://www.rfc-editor.org/info/rfc5733>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.

11. Informative References

- [XMLCOM-WEB] XML.com, "Converting Between XML and JSON", 2006, <<https://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html>>.

Appendix A. Appendix A. Media Type Registration: application/epp+json

MIME media type name: application

MIME subtype name: epp+json

Required parameters: none

Optional parameters: Same as the charset parameter of application/json as specified in [RFC8259].

Encoding considerations: Same as the encoding considerations of application/xml as specified in [RFC8259].

Security considerations: This type has all of the security considerations described in [RFC8259] plus the considerations specified in the Security Considerations section of this document.

Published specification: This document.

Applications that use this media type: RESTful EPP client and server implementations.

Additional information: None

Magic number(s): None.

File extension(s): .json

Macintosh file type code(s): "TEXT"

Person & email address for further information: See the "Author's Address" section of this document.

Intended usage: COMMON

Author/Change controller: IETF

Authors' Addresses

Maarten Wullink

SIDN Labs

Email: maarten.wullink@sidn.nl

URI: <https://sidn.nl/>

Marco Davids

SIDN Labs

Email: marco.davids@sidn.nl

URI: <https://sidn.nl/>