Workgroup:	Network Working Group
Internet-Draft:	draft-wullink-restful-epp-json-00
Published:	15 January 2024
Intended Status:	Standards Track
Expires:	18 July 2024
Authors:	M. Wullink *SIDN Labs*	M. Davids *SIDN Labs*
---	---	---

XML to JSON conversion rules for RESTful EPP

Abstract

This document describes the rules for converting an EPP [RFC5730] XML message to a JSON [RFC8259] message for use with 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 18 July 2024.

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
3. Conventions Used in This Document 3
4. Conversion Rules 4
   4.1. Empty 4
   4.2. Pure text content 4
   4.3. Attributes only 5
   4.4. Pure text content and attributes 5
   4.5. Child elements with different names 6
   4.6. Child elements with identical names 6
   4.7. Child elements and contiguous text 7
5. Examples 8
   5.1. Hello 8
   5.2. Login 10
   5.3. Logout 10
   5.4. Check 10
   5.5. Info 10
   5.6. Poll 12
   5.7. Poll Ack 14
   5.8. Transfer Query 14
   5.9. Create 15
   5.10. Delete 16
   5.11. Renew 17
   5.12. Transfer Request 18
   5.13. Transfer Cancel 18
   5.14. Transfer Reject 19
   5.15. Transfer Approve 20
   5.16. Update 20

6. IANA Considerations 21
7. Internationalization Considerations 21
8. Security Considerations 21
9. Acknowledgments 21
10. References 21
10.1. Normative References 21
10.2. Informative References 22
Appendix A. Appendix A. Media Type Registration: application/epp+json 22
Authors' Addresses 23

1. Introduction

The Extensible Provisioning Protocol (EPP) [RFC5730] uses an XML based protocol. The schemas for validating EPP XML messages are published as part of the EPP RFCs.

RESTful EPP (REPP), however has suport for multiple data formats such as the JavaScript Object Notation (JSON) Data Interchange Format [RFC8259].

This document describes the rules for converting a valid EPP XML message to JSON message, which can be used with REPP.

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].

Stateful EPP - The definition according to Section 2 of [RFC5730].

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

In general a single XML element allows for the following forms

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 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 atributes 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 atributes 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

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:

XML namespaces are not converted to JSON and are ignored.

```
{
   "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 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:registrant>jd1234</domain:registrant>
            <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 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 Transfer Query request message is not used for REPP.

Example Transfer Query response:

```
S: HTTP/2 200 OK
S: Date: Fri, 17 Nov 2023 12:00:00 UTC
S: Server: Example REPP server v1.0
S: Content-Length: 230
S: Content-Type: application/epp+xml
S: Content-Language: en
S: REPP-Eppcode: 1000
S:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <!-- The rest of the response is omitted here -->
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>XYZ-12345</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.9. Create

Example Domain Create request:

```
C: POST /repp/v1/domains HTTP/2
C: Host: repp.example.nl
C: Authorization: Bearer <token>
C: Accept: application/epp+xml
C: Content-Type: application/epp+xml
C: REPP-Svcs: urn:ietf:params:xml:ns:domain-1.0
C: Accept-Language: en
C: Content-Length: 220
C:
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <domain:create
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.nl</domain:name>
C:        <!-- The rest of the request is omitted here -->
C:      </domain:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example Domain Create response:

```
S: HTTP/2 200
S: Date: Fri, 17 Nov 2023 12:00:00 UTC
S: Server: Example REPP server v1.0
S: Content-Language: en
S: Content-Length: 642
S: Content-Type: application/epp+xml
S: Location: https://repp.example.nl/repp/v1/domains/example.nl
S: REPP-Eppcode: 1000
S:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
S:     xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
S:   <response>
S:     <result code="1000">
S:       <msg>Command completed successfully</msg>
S:     </result>
S:     <resData>
S:       <domain:creData>
S:         <!-- The rest of the response is omitted here -->
S:       </domain:creData>
S:     </resData>
S:     <trID>
S:       <clTRID>ABC-12345</clTRID>
S:       <svTRID>54321-XYZ</svTRID>
S:     </trID>
S:   </response>
S:</epp>
```

5.10. Delete

The Delete request message is not used for REPP.

Example Domain Delete response:

```
S: HTTP/2 200 OK
S: Date: Fri, 17 Nov 2023 12:00:00 UTC
S: Server: Example REPP server v1.0
S: Content-Length: 80
S: REPP-Svtrid: XYZ-12345
S: REPP-Cltrid: ABC-12345
S: REPP-Eppcode: 1000
S:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>XYZ-12345</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.11. Renew

The Renew request message is not used for REPP.

Example Renew response:

```
S: HTTP/2 200 OK
S: Date: Fri, 17 Nov 2023 12:00:00 UTC
S: Server: Example REPP server v1.0
S: Content-Language: en
S: Content-Length: 205
S: Location: https://repp.example.nl/repp/v1/domains/example.nl
S: Content-Type: application/epp+xml
S: REPP-Eppcode: 1000
S:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <resData>
S:      <!-- The rest of the response is omitted here -->
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>XYZ-12345</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.12. Transfer Request

The Transfer request message is not used for REPP.

Example Transfer response:

```
S: HTTP/2 200 OK
S: Date: Fri, 17 Nov 2023 12:00:00 UTC
S: Server: Example REPP server v1.0
S: Content-Language: en
S: Content-Length: 328
S: Content-Type: application/epp+xml
S: Location: https://repp.example.nl/repp/v1/domains/example.nl/transfers/latest
S: REPP-Eppcode: 1001
S:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1001">
S:      <msg>Command completed successfully; action pending</msg>
S:    </result>
S:    <resData>
S:      <!-- The rest of the response is omitted here -->
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>XYZ-12345</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.13. Transfer Cancel

The Transfer Cancel request message is not used for REPP.

Example response:

```
S: HTTP/2 200 OK
S: Date: Fri, 17 Nov 2023 12:00:00 UTC
S: Server: Example REPP server v1.0
S: Content-Length: 80
S: REPP-Svtrid: XYZ-12345
S: REPP-Cltrid: ABC-12345
S: REPP-Eppcode: 1000
S:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>XYZ-12345</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.14. Transfer Reject

The Transfer Reject request message is not used for REPP.

Example Reject response:

```
S: HTTP/2 200 OK
S: Date: Fri, 17 Nov 2023 12:00:00 UTC
S: Server: Example REPP server v1.0
S: Content-Length: 80
S: REPP-Svtrid: XYZ-12345
S: REPP-Cltrid: ABC-12345
S: REPP-Eppcode: 1000
S:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>XYZ-12345</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.15. Transfer Approve

The Transfer Approve request message is not used for REPP.

Example Approve response:

```
S: HTTP/2 200 OK
S: Date: Fri, 17 Nov 2023 12:00:00 UTC
S: Server: Example REPP server v1.0
S: Content-Length: 80
S: REPP-Svtrid: XYZ-12345
S: REPP-Cltrid: ABC-12345
S: REPP-Eppcode: 1000
S:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>XYZ-12345</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

5.16. Update

Example request:

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <domain:update
C:       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
C:        <domain:name>example.nl</domain:name>
C:          <!-- The rest of the request is omitted here -->
C:      </domain:update>
C:    </update>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Example response:

```
S: HTTP/2 200 OK
S: Date: Fri, 17 Nov 2023 12:00:00 UTC
S: Server: Example REPP server v1.0
S: Content-Length: 80
S: REPP-Svtrid: XYZ-12345
S: REPP-Cltrid: ABC-12345
S: REPP-Eppcode: 1000
S:
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Command completed successfully</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>XYZ-12345</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

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. References

10.1. 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>.

10.2. 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/