Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 15, 2016

A. Schiltknecht, Ed.
Viagenie

August 14, 2015

A JSON format for LGR files
draft-schiltknecht-lgr-json-00

Abstract

This document defines a JSON format for LGRs (Label Generation Rules). LGRs are used to represent rules for validating identifier labels and their alternate representations. These LGRs are expressed in XML as defined in [I-D.ietf-lager-specification].

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 http://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 February 15, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document describes a JSON format for representing LGRs as described in [I-D.ietf-lager-specification].

The key design considerations taken into account in this document are

- Round-tripping (converting an XML LGR to JSON and back) will yield the same semantic result as the starting point. All XML elements, attributes and values are guaranteed to be preserved.

- The ordering of elements MUST be preserved, as it is of importance in the original LGR XML specification.

The terms "JSON object", "JSON array", "JSON member", and "JSON value" are to be interpreted as described in [RFC7159].

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 RFC 2119 [RFC2119].

2. Converting from XML to JSON

This section explains how to convert an XML LGR to JSON, by defining a simple mapping between XML nodes and JSON objects.

2.1. Basic structure

As a valid JSON object, the basic layout of an LGR in JSON is as follows:

```
{
    "meta": [
        ...
    ],
    "data": [
        ...
    ],
    "rules": [
        ...
    ]
}
```

As expressed in [I-D.ietf-lager-specification], only the "data" object is mandatory.

The conversion scheme follows these general conventions:

- Each XML element is a JSON array with 3 members: the name of the element (called "name"), a (potentially empty) JSON object representing the attributes of the element (called "attributes"), and the value of the element (called "value").

- The type of the "value" element (last element of the JSON array) depends on the context, and will be defined in each of the following sections.

The use of JSON arrays which are ordered sequences allows to keep the order of the declarations from the XML.

For example, the XML extract:

```
<char cp="0063"/>
```

will be converted to

```
["char", {"cp": "0063"}, []]
```

2.2. Metadata

The type of the "value" is a string, except for the "references" element where an array of the JSON-converted "reference" child XML elements is used.

Given the following XML "meta" element:

```
   <meta>
       <version comment="initial version">1</version>
       <date>2010-01-01</date>
       <language>sv</language>
       <scope type="domain">example</scope>
       <validity-start>2010-01-01</validity-start>
       <validity-end>2013-12-31</validity-end>
       <description type="text/html">
           <![CDATA[
           This language table was developed with the
           <a href="http://swedish.example/">Swedish
           examples institute</a>.
           ]]>
       </description>
       <description>
       <unicode-version>6.3.0</unicode-version>
       <references>
         <reference id="0" comment="the most recent" >The
               Unicode Standard 6.2</reference>
         <reference id="1" >RFC 5892</reference>
         <reference id="2" >Big-5: Computer Chinese Glyph
            and Character Code Mapping Table, Technical Report
            C-26, 1984</reference>
       </references>
    </meta>
```

the converted JSON "meta" array is:

```
"meta":
[
    ["version", {"comment": "initial version"}, "1"],
    ["date", {}, "2010-01-01"],
    ["language", {}, "sv"],
    ["scope", {"type": "domain"}, "example"],
    ["validity-start", {}, "2010-01-01"],
    ["validity-end", {}, "2013-12-31"],
    ["description", {"type": "text/html"}, "This language table was developed
with the <a href=\"http://swedish.example/\">Swedish examples institute</a>."],
    ["unicode-version", {}, "6.3.0"],
    ["references", {}, [
        ["reference", {"id": "0", "comment": "the most recent"}, "The Unicode
Standard 6.2"],
        ["reference", {"id": "1"}, "RFC 5892"],
        ["reference", {"id": "2"}, "Big-5: Computer Chinese Glyph and Character
Code Mapping Table, Technical Report C-26, 1984"]
    ]]
]
```

2.3. Code Points and variants

All code point data is contained in the "data" section of an LGR. There are two types of data:

- Code points ("char" elements), which can have variants.
- Range of code points ("range" elements), defined by their first and last code point, and cannot have variants.

As a consequence, the type of the "value" is an array, containing the variants of a "char" elements. For variants, it is an empty array.

Typical conversions are described in the following examples:

```
<data>
    <char cp="002D"/>
    <range first-cp="0030" last-cp="0039"/>
    <char cp="006C 00B7 006C" comment="Catalan middle dot"/>
</data>
```

```
"data":
[
    ["char", {"cp": "002D"}, []],
    ["range", {"first-cp": "0030", "last-cp": "0039"}, []],
    ["char", {"cp": "006C 00B7 006C", "comment": "Catalan middle dot"}, []]
]
```

For variants:

```
<char cp="00F6">
    <var cp="006F 0065" type="block"/>
</char>
```

```
["char", {"cp": "00F6"}, [
    ["var", {"cp": "006F 0065", "type": "block"}, []]
]]
```

2.4. Whole Label Rules and actions

Rules, classes and actions are defined in the "rules" section of an LGR, as an array of JSON objets.

The "value" element will have the following types:

- For rules, an array of the rule's match operators.

- For classes, an array of the class' codepoints, or a string if the shorthand notation is used.

- For actions, an empty array since actions do not have any value or child.

```
    <rules>
      <rule name="catalan-middle-dot" ref="0">
          <look-behind>
              <char cp="006C" />
          </look-behind>
          <anchor />
          <look-ahead>
              <char cp="006C" />
          </look-ahead>
      </rule>

      <class name="virama" property="ccc:9" />
      <rule name="joiner"  ref="1" >
          <look-behind>
              <class by-ref="virama" />
          </look-behind>
          <anchor />
      </rule>

      <difference name="consonants">
           <class comment="all letters">0061-007A</class>
           <class comment="all vowels">
                   0061 0065 0069 006F 0075
           </class>
       </difference>

       <rule name="three-or-more-consonants">
           <start />
           <class by-ref="consonants" count="3+" />
           <end />
      </rule>

      <rule name="non-preferred"
            comment="matches any non-preferred code point">
          <complement comment="non-preferred" >
              <class from-tag="preferred" />
          </complement>
      </rule>

      <action disp="consonants"
              match="three-or-more-consonants" />
      <action disp="block" any-variant="block" />
      <action disp="activate" all-variants="allocate"
              not-match="non-preferred" />
    </rules>

  "rules":
  [
```

```
    [
        "rule", {
            "name": "catalan-middle-dot",
            "ref": "0"
        },
        [
            [
                "look-behind", {},
                [
                    ["char", {"cp": "006C"}, []]
                ]
            ],
            [
                "anchor", {}, []
            ],
            [
                "look-ahead", {},
                [
                    ["char", {"cp": "006C"}, []]
                ]
            ]
        ]
    ],

    [
        "class",
        {
            "name": "virama",
            "property": "ccc:9"
        },
        []
    ],
    [
        "rule",
        {
            "name": "joiner",
            "ref": "1"
        },
        [
            [
                "look-behind",
                {},
                [
                    [
                        "class",
                        {"by-ref": "virama"},
                        []
                    ]
```

```
                ]
            ],
            ["anchor", {}, []]
        ]
    ],

    [
        "difference",
        {"name": "consonants"},
        [
            [
                "class",
                {"comment": "all letters"},
                "0061-007A"
            ],
            [
                "class",
                {"comment": "all vowels"},
                "0061 0065 0069 006F 0075"
            ]
        ]
    ],

    [
        "rule",
        {"name": "three-or-more-consonants"},
        [
            [
                "start", {}, []
            ],
            [
                "class",
                {
                    "by-ref": "consonants",
                    "count": "3+"
                },
                []
            ],
            [
                "end", {}, []
            ]
        ]
    ],

    [
        "rule",
        {
            "name": "non-preferred",
```

```
                "comment": "matches any non-prefered code point"
            },
            [
                [
                    "complement",
                    {"comment": "non-preferred"},
                    [
                        [
                            "class",
                            {"from-tag": "preferred"},
                            []
                        ]
                    ]
                ]
            ]
        ],

        [
            "action",
            {
                "disp": "consonants",
                "match": "three-or-more-consonants"
            },
            []
        ],
        [
            "action",
            {
                "disp": "block",
                "any-variant": "block"
            },
            []
        ],
        [
            "action",
            {
                "disp": "activate",
                "all-variants": "allocate",
                "not-match": "non-preferred"
            },
            []
        ]
    ]
```

3. Converting from JSON to XML

When converting a JSON LGR to XML format, proper escaping of text content MUST be done.

An empty "value" (empty list, empty object or empty string) is an XML element without value nor child.

4. Acknowledgements

TODO

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

Since JSON is used as a format, the security risks discussed in [RFC7159] are to be considered.

7. Normative References

[I-D.ietf-lager-specification]
Davies, K. and A. Freytag, "Representing Label Generation Rulesets using XML", draft-ietf-lager-specification-00 (work in progress), August 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

Appendix A. ABNF syntax

TODO

Author's Address

Audric Schiltknecht (editor)
Viagenie
246 Aberdeen
Quebec, QC G1R 2E1
Canada

Email: audric.schiltknecht@viagenie.ca
URI: http://viagenie.ca