Document Number: DSP0211

Date: 2012-08-28

Version: 1.0.0a

CIM-RS Payload Representation in JSON

Information for Work-in-Progress version:

This specification is not a standard. It does not necessarily reflect the views of the DMTF or all of its members. Because this document is a Work in Progress, this specification may still change, perhaps profoundly. This document is available for public review and comment until the stated expiration date.

This document expires on: 2013-02-28

Provide any comments through the DMTF Feedback Portal: http://www.dmtf.org/standards/feedback

Document Type: Specification

Document Status: Work in Progress

Document Language: en-US

CONTENTS

Tables

Foreword

The *CIM-RS Payload Representation in JSON* (DSP0211) specification was prepared by the DMTF CIM-RS Working Group, based on work of the DMTF CIM-RS Incubator.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

Acknowledgments

The DMTF acknowledges the following individuals for their contributions to this document:

- Cornelia Davis, EMC
- George Ericson, EMC
- Johannes Holzer, IBM
- Robert Kieninger, IBM
- Wojtek Kozaczynski, Microsoft
- Larry Lamers, VMware
- Andreas Maier, IBM (editor)
- Bob Tillman, EMC
- Marvin Waschke, CA Technologies

Introduction

The information in this document should be sufficient to unambiguously identify the representation of the payload elements defined in DSP0210, in JSON (JavaScript Object Notation).

The target audience for this specification is typically implementers who are writing WBEM servers, clients, or listeners supporting the CIM-RS protocol with a payload representation in JSON.

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

- Document titles are marked in *italics*.
- ABNF rules and JSON text are in `monospaced font`.

ABNF usage conventions

Format definitions in this document are specified using ABNF (see RFC5234), with the following deviations:

- Literal strings are to be interpreted as case-sensitive UCS characters, as opposed to the definition in RFC5234 that interprets literal strings as case-insensitive US-ASCII characters.

CIM-RS Payload Representation in JSON

1 Scope

This specification is a payload representation specification for the CIM-RS protocol defined in DSP0210, describing a representation of CIM-RS payload elements in JSON (JavaScript Object Notation, see ECMA-262).

Specifically, it describes how the abstract payload elements defined in DSP0210 are represented in JSON and how a JSON representation of these payload elements is identified using an Internet media type.

Background information for CIM-RS is described in a white paper, DSP2032.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

ANSI/IEEE 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*, August 1985, http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=30711

DMTF DSP0004, *CIM Infrastructure Specification 2.7*, http://www.dmtf.org/standards/published_documents/DSP0004_2.7.pdf

DMTF DSP0210, *CIM-RS Protocol 1.0*, http://www.dmtf.org/standards/published_documents/DSP0210_1.0.pdf

DMTF DSP0223, *Generic Operations 1.0*, http://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf

ECMA-262, *ECMAScript Language Specification, Edition 5.1*, June 2011, http://www.ecma-international.org/publications/standards/Ecma-262.htm

IETF RFC4627 (Informational), *The application/json Media Type for JavaScript Object Notation (JSON)*, July 2006, http://tools.ietf.org/html/rfc4627

IETF RFC5234, *Augmented BNF for Syntax Specifications: ABNF*, January 2008, http://tools.ietf.org/html/rfc5234

ISO/IEC 10646:2003, *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*, http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921_ISO_IEC_10646_2003(E).zip

ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards (2004, 5th edition)*, http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse

The Unicode Consortium, *The Unicode Standard, Version 5.2.0, Annex #15: Unicode Normalization Forms*, http://www.unicode.org/reports/tr15/

3 Terms and definitions

In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"), "may", "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Annex H. The terms in parenthesis are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 5.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The terms defined in DSP0004, DSP0223, and DSP0210 apply to this document. Specifically, this document uses the terms "namespace", "qualifier", "qualifier type", "class", “creation class”, "ordinary class", "association", "indication", "instance", "property", "ordinary property", "reference", "method", "parameter", and "return value" defined in DSP0004.

This document does not define additional terms; some terms defined in these documents are repeated for convenience.

3.1

CIM-RS payload element

a particular type of content of the entity body of the HTTP messages used by the CIM-RS protocol. Payload elements are abstractly defined in DSP0210, and concretely in CIM-RS payload representation specifications, such as this document.

3.2

CIM-RS payload representation

an encoding format that defines how the abstract payload elements defined in DSP0210 are encoded in the entity body of the HTTP messages used by the CIM-RS protocol. This includes resource representations.

3.3

CIM-RS payload representation specification

a specification that defines a CIM-RS payload representation, such as this document.

3.4

CIM-RS protocol

the RESTful protocol defined in DSP0210, for which this document describes a payload representation in JSON.

3.1

CIM-RS resource

an entity in a WBEM server or WBEM listener that can be referenced using a CIM-RS resource identifier and thus can be the target of an HTTP method in the CIM-RS protocol. Also called "resource" in this document.

3.2

CIM-RS resource identifier
a URI that is a reference to a CIM-RS resource in a WBEM server or WBEM listener, as defined in DSP0210. Also called "resource identifier" in this document.

3.3
Internet media type
a string identification for representation formats in Internet protocols. Originally defined for email attachments and termed "MIME type". Because the CIM-RS protocol is based on HTTP, it uses the definition of media types from section 3.7 of RFC2616.

3.4
Normalization Form C
a normalization form for UCS characters that avoids the use of combining marks where possible and that allows comparing UCS character strings on a per-code-point basis. It is defined in The Unicode Standard, Annex #15.

3.1
resource representation
a representation of a resource or some aspect thereof, in some format. A particular resource may have any number of representations. The format of a resource representation is identified by a media type. In the CIM-RS protocol, the more general term "payload representation" is used, because not all payload elements are resource representations.

3.2
UCS character
a character from the Universal Character Set defined in ISO/IEC 10646:2003. See DSP0004 for the usage of UCS characters in CIM strings. An alternative term is "Unicode character".

3.3
UCS code position
a numeric identification for a UCS character in the range of 0x0 to 0x10FFFF, as defined in ISO/IEC 10646:2003.

3.4
WBEM client
the client role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see DSP0210.

3.5
WBEM listener
the event listener role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see DSP0210.

3.6
WBEM server
the server role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see DSP0210.

4 Symbols and abbreviated terms

The abbreviations defined in DSP0004, DSP0223, and DSP0210 apply to this document. The following additional abbreviations are used in this document.

4.1

ABNF
Augmented Backus-Naur Form, as defined in RFC5234.

4.2
CIM
Common Information Model, as defined by DMTF.

4.3
CIM-RS
CIM RESTful Services
The RESTful protocol for CIM defined in this document and related documents.

4.4
ECMAScript
a scripting language that is the standard version of what was called JavaScript. It is defined in ECMA-262.

4.5
IANA
Internet Assigned Numbers Authority; see http://www.iana.org.

4.6
JSON
JavaScript Object Notation, as defined in ECMA-262.

4.7
REST
Representational State Transfer, as originally and informally described in *Architectural Styles and the Design of Network-based Software Architectures*.

4.8
UCS
Universal Character Set, as defined in ISO/IEC 10646:2003.

4.9
URI
Uniform Resource Identifier, as defined in RFC3986.

4.10
UTF-8
UCS Transformation Format 8, as defined in ISO/IEC 10646:2003.

4.11
WBEM
Web Based Enterprise Management, as defined by DMTF.

5 Conformance

A representation of CIM-RS payload elements in JSON conforms to this document only if it conforms to all normative rules stated in this document.

The term "CIM-RS representation in JSON" shall be used only for representations of CIM-RS payload elements in JSON that conform to this document.

6 CIM-RS payload representation in JSON

This clause defines the representation of the CIM-RS payload in JSON.

The JSON grammar is defined in clause 15.12.1 ("The JSON Grammar") of ECMA-262. Care should be taken to distinguish text in ECMA-262 that applies to the JSON grammar from text that applies to the ECMAScript (formerly: JavaScript) language. However, text in ECMA-262 outside of its clause 15.12.1 but referenced from within that clause applies unless otherwise noted in this document.

Note that although RFC4627 defines the grammar of the JSON language consistently with clause 15.12.1 of ECMA-262, RFC4627 is an informational RFC whose purpose is to describe the Internet media type for JSON but not to be the normative definition of the JSON grammar. For this reason, this document references ECMA-262 as the normative definition of the JSON grammar, but yet references RFC4627 where needed.

6.1 Overview

This subclause describes informally and at a high level how the CIM-RS payload elements defined in DSP0210 are represented in JSON.

CIM-RS payload elements are represented as JSON objects. The attributes of these JSON objects match the properties of the payload elements 1:1, in name, datatype and meaning. Nested elements in these payload elements are represented as nested JSON objects. Arrays in these payload elements are represented as JSON arrays. For details, see 6.3.

The Internet media type identifying the JSON representation of CIM-RS is the standard media type registered for JSON at IANA (application/json). For details, see 6.2.5.

Defining a new media type specific for the CIM-RS representation of JSON was considered and dismissed, because the value of using a well-known and broadly supported standard media type was deemed higher than the advantage of being able to distinguish JSON for representing CIM-RS from general JSON, or multiple flavors of JSON for representing CIM-RS from each other. Multiple incompatible flavors of JSON representations of CIM-RS can be distinguished with the same media type by using different major version numbers in the version parameter of the media type.

6.2 General requirements

6.2.1 Conformance to the JSON grammar

CIM-RS payload elements represented in JSON shall conform to the grammar defined by the symbol *JSONText* defined in clause 15.12.1 ("The JSON Grammar") of ECMA-262.

6.2.2 Whitespace

ECMA-262 defines what the set of whitespace characters for JSON is (different from the set of whitespace characters for ECMAScript), but it does not explicitly state whether the whitespace usage rules for ECMAScript also apply to JSON.

CIM-RS payload elements represented in JSON shall conform to the rules for whitespace as defined in subclause 7.2 (White Space) of ECMA-262.

6.2.3 Character repertoire, representation, encoding and escaping

The JSON grammar defined in clause 15.12.1 of ECMA-262 references the *SourceCharacter* symbol defined in its clause 6 as the basis for the characters of its grammar be it for identifiers, delimiters or values. The definition of the *SourceCharacter* symbol applies to the ECMAScript use of JSON and uses

the character repertoire of Unicode V3, requires a representation of UCS characters in Normalization Form C, and effectively implies a requirement for an encoding in UTF-16 (or one of its little endian and big endian derivatives).

The following rules apply to a use of the *SourceCharacter* symbol for the representation of CIM-RS payload elements in JSON:

1) The character repertoire of *SourceCharacter* shall be that defined for values of the CIM string type (defined in DSP0004). Note that this character repertoire is larger than the character repertoire defined by ECMA-262.

2) *SourceCharacter* shall be represented in Normalization Form C.

3) *SourceCharacter* shall be encoded in UTF-8. As a consequence, the entire payload element will be encoded in UTF-8, and that character encoding is therefore not being indicated in the CIM-RS payload or in any HTTP header fields.

ECMA-262 defines backslash-based escaping for the representation of UCS characters, using their UCS code positions. However, in the definition of the *UnicodeEscapeSequence* symbol in its clause 7.8.4 ("String Literals"), ECMA-262 limits the representation of UCS code positions to a value range of four hex digits. This is not sufficient for representing the character repertoire defined for values of the CIM string type (it is also not sufficient for representing the character repertoire used by ECMA-262 itself).

Therefore, the representation of CIM-RS payload elements in JSON shall support the following extended definition of the *UnicodeEscapeSequence* symbol:

```
UnicodeEscapeSequence ::
    u HexDigit HexDigit HexDigit HexDigit
    u HexDigit HexDigit HexDigit HexDigit HexDigit
    u HexDigit HexDigit HexDigit HexDigit HexDigit HexDigit
```

NOTE: This extended definition is consistent with the four-to-six-digit form of the short identifier for UCS characters defined in clause 6.5 of ISO/IEC 10646:2003 (for example, U+000A, U+12345, and U+10FFFF).

ECMA-262 defines backslash-based escaping for a number of popular characters, e.g. "\n". It states their escape sequences, but it misses to define what they stand for. RFC4627 does define both the escape sequences and what they stand for.

6.2.4 Version of the payload representation in JSON

DSP0210 requires that CIM-RS payload representation specifications define a version for the payload representations they define.

The full version (m.n.u) of this document, without any draft levels, shall be used to identify the full version of the JSON payload representation.

6.2.5 Internet media type

DSP0210 requires that CIM-RS payload representation specifications define a unique Internet media type that identifies the representation.

Only the standard media type for JSON defined in RFC4627 ("application/json") shall be used to identify the representation of CIM-RS payload elements in JSON defined in this document. This media type is registered with IANA (see *IANA MIME Media Types*).

Note that DSP0210 defines requirements for specifying parameters on media types that identify the representation of CIM-RS payload elements. One example for such a parameter is "version", specifying the version of the payload representation.

Therefore, the media type identifying version 1.0.0 of the JSON representation would be:

```
application/json; version=1.0.0
```

6.2.6 Representation of CIM-RS abstract datatypes

This subclause defines how values of the abstract datatypes used in DSP0210 for the definition of the attributes of the abstract payload elements are represented in JSON.

Table 1 lists the abstract datatypes and their mapping to JSON datatypes.

Table 1 – Representation of CIM-RS abstract datatypes in JSON

Abstract Datatype	JSON Datatype	Additional Rules
String	string	See 6.2.3 for requirements on escaping and encoding
Integer	number	
ElementValue	object member	Values of CIM elements (that is, properties, parameters, return values) shall be represented in JSON as described in 6.2.7
MethodLink	object member	Method invocation links shall be represented in JSON as described in 6.2.11
URI	string	The string value shall be the CIM-RS resource identifier of the referenced resource in any valid format (see DSP0210).
Instance	object	See 6.3.4

6.2.7 Representation of CIM element values

Values of CIM elements (that is, properties, parameters, return values) shall be represented in JSON as follows:

The element value is represented as a JSON object member, where the name of the object member is the name of the CIM element; and the value of the JSON object member is a representation of the CIM element value as defined in Table 2, using the indicated JSON datatype.

The CIM datatype of the element (that is, the "type" child attribute of the ElementValue datatype) is intentionally not represented, for simplicity. It is expected that the JSON representation of CIM-RS is used by environments with simple and possibly dynamic type systems (such as JavaScript or Python), without a need to represent the elements using strong typing based on the CIM datatypes.

Table 2 – Representation of CIM datatypes in JSON

CIM Datatype	JSON Datatype	Additional Rules
boolean	boolean	Note that JSON is case sensitive w.r.t. the literals true and false
string	string	See 6.2.3 for requirements on escaping and encoding
char16	string	See 6.2.3 for requirements on escaping and encoding
string, with OctetString qualifier	string	Shall be represented as if it was a normal CIM string-typed value (that is, without the OctetString qualifier)
uint8[], with OctetString qualifier	number array	Shall be represented as if it was a normal uint8-array-typed value (that is, without the OctetString qualifier)

CIM Datatype	JSON Datatype	Additional Rules
string, with EmbeddedInstance qualifier	object	The embedded instance shall be represented as a JSON object as defined in 6.3.4. Its *class* attribute shall be the name of the creation class of the embedded instance. Note that the creation class may differ from the class specified in the value of the EmbeddedInstance qualifier.
string, with EmbeddedObject qualifier	object or null value	If the embedded object is an instance, it shall be represented as a JSON object as defined in 6.3.4. Its *class* attribute shall be the name of the creation class of the embedded instance. If the embedded object is a class, it shall not be represented and instead the JSON null value shall be represented.
datetime	string	The string value shall be the 25-character datetime string defined in DSP0004
uint8,16,32,64	number	
sint8,16,32,64	number	
real32,64	number or string	See 6.2.8
<classname> ref	string	See 6.2.9
array of any CIM type	array of corresponding JSON type	The type string shall reflect the type of the array entries

Examples for representing named CIM elements (that is, properties or parameters) of these datatypes:

```
. . .
  "ABoolean": true,
  "AString": "some text",
  "AChar16": "Z",
  "AnOctetstringViaString": "0x00000007616263",
  "AnOctetstringViaUint8Array": [ 0, 0, 0, 7, 0x61, 0x62, 0x63 ],
  "AnEmbeddedInstance": {
    "kind": "instance",
    "class": "CIM_ComputerSystem",
    "properties": {
      "InstanceID": "sys:1",
      "ElementName": "system #1" }
  },
  "AnEmbeddedObjectThatIsAnInstance": {
    "kind": "instance",
    "class": "CIM_ComputerSystem",
    "properties": {
      "InstanceID": "sys:1",
      "ElementName": "system #1" }
  },
  "ADatetime": "20120213175830.123456+060",
  "AUint16": 20000,
  "ASint16": -16000,
  "AReal32": 3.1415927,
  "ARef": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1",
  "ABooleanArray": [ true, false, true ],
```

```
  "AStringArray": [ "some text", null, "more text\n" ],
  "AReal64Array": [ 1E-42, NaN, "-Infinity" ],
. . .
```

6.2.8 Representation of CIM real32 and real64 datatypes

The CIM real32 and real64 types are based on the IEEE 754 Single and Double formats (see DSP0004); values of these types shall be represented in JSON as follows, depending on their value:

- the IEEE special values *positive infinity*, *negative infinity*, and *not-a-number* shall be represented as JSON string-typed values using the following strings:

  positive infinity: `"Infinity"`

  negative infinity: `"-Infinity"`

  not-a-number: `"NaN"`

- any other values are normal floating point numbers and shall be represented as JSON number-typed values, using a precision for the significand of at least 9 decimal digits for real32 and at least 17 digits for real64.

NOTE: The strings used for representing the IEEE special values are consistent with Python's serialization of float-typed values in JSON, and with Java's serialization of float-typed values as strings. These strings are not consistent with the representation of the special values in the XML datatypes xs:float and xs:double.

NOTE: JSON numbers only support lexical notations with a basis of 10 (e.g. 4.56E-3). The value space of CIM real32 and real64 typed values is defined by the IEEE 754 Single and Double formats, which have a basis of 2. The definition of a minimum precision for the significand guarantees that the value of CIM real types does not change when converting it back and forth between the (10-based) JSON representation and a (2-based) internal representation (see subclause 5.6 in IEEE 754).

Examples:

```
. . .
  "Throughput": 3.45E3,
  "ErrorRate": "NaN",
. . .
```

6.2.9 Representation of CIM reference datatypes

Values of CIM reference-typed elements (that is, declared as <classname> ref) shall be represented in JSON such that the JSON value is the CIM-RS resource identifier of the referenced instance in any valid format defined in DSP0210.

The class declared in the reference is not represented, again for simplicity of the JSON representation.

Example for a reference property named System that is declared as type "CIM_ComputerSystem ref":

```
. . .
  "System": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1"
. . .
```

6.2.10 Representation of CIM Null values

CIM Null values shall be represented using the JSON literal *null*.

433 Note that the JSON literal null is case sensitive.

434 Example:

```
435 . . .
436   "ElementName": null,
437   "PossibleStates": [1, null, 3],
438 . . .
```

439 6.2.11 Representation of method invocation links

440 Method invocation links shall be represented in JSON as a JSON object member, where the name of the
441 object member is the name of the CIM method (without any parenthesis or parameters); and the value of
442 the JSON object member is a JSON String typed value that is the resource identifier of the method
443 invocation resource.

444 Example:

```
445 . . .
446   "RequestStateChange":
447     "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1/RequestStateChange"
448 . . .
```

449 6.3 Representation of protocol payload elements

450 This subclause defines how the CIM-RS payload elements defined in DSP0210 are represented in JSON.

451 Table 3 lists the payload elements defined in DSP0210.

452 **Table 3 – CIM-RS payload elements**

Payload Element	Meaning	Description
ServerEntryPoint	representation of the server entry point resource of a CIM-RS server, describing protocol-level capabilities of the server, and providing resource identifiers for performing global operations	See 6.3.2
ListenerEntryPoint	representation of the listener entry point resource of a CIM-RS listener, describing protocol-level capabilities of the listener	See 6.3.3
Instance	representation of a CIM instance; that is, a CIM-modeled resource, representing an aspect of a managed object in the managed environment	See 6.3.4
InstanceCollection	representation of a set of CIM instances in an instance collection	See 6.3.5
MethodRequest	the data used to request the invocation of a method	See 6.3.6
MethodResponse	the data used in the response of the invocation of a method	See 6.3.7
IndicationDeliveryRequest	the data used to request the delivery of an indication to a listener destination	See 6.3.8
ErrorResponse	the data used in an error response to any request	See 6.3.9

453 6.3.1 Format of payload element descriptions

454 The descriptions in the following subclauses use a lightweight approach for defining the JSON structure
455 for the various payload elements. The following example illustrates this description approach:

```
456 {
```

```
  "kind": "instance",
  "self": (value),
  "class": (value),
  "properties": {
    (value): (value)#
  },?
  "methods": {
    (value): (value)#
  }?
}
```

All text in such a description is to be understood literally as stated, except for whitespace characters used before and after JSON tokens (see 6.2.2), and except for the following special indicators:

# indicates that the JSON object member or JSON array element to the left of the # may be present zero or more times in a comma-separated list.

? indicates that the JSON object member or JSON array element to the left of the ***?*** may be present or absent.

(value) is replaced with a JSON value, according to the description in the respective subclause. The literal inside of the parenthesis is typically more specific than "value".

Note that the rules on using # and ***?*** are not precise w.r.t. the presence of commas delimiting JSON object members or JSON array elements. However, the presence of commas results from the general JSON syntax rules; that is, exactly one comma is required between members or elements, and no trailing comma is permitted after the last member or element.

An example for a valid payload element conforming to the example description above would be:

```
{
  "kind": "instance",
  "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0",
  "class": "CIM_RegisteredProfile",
  "properties": {
    "InstanceID": "DMTF:Fan:1.1.0",
    "RegisteredName": "Fan",,
    "RegisteredOrganization": 2,
    "RegisteredVersion": "1.1.0"
  }
}
```

6.3.2 ServerEntryPoint payload element

ServerEntryPoint payload elements (see DSP0210) shall be represented using the following JSON structure:

```
{
  "kind": "serverentrypoint"
  "self": (self),
  "namespaces": [
    { "name": (name),
      "enumeration": (enumeration),
      "creation": (creation),
```

```
      "staticmethods": [
        (static-method-name): (static-method-uri)#
      ],?
      "protocolversions": [
        (protocolversion)#
      ],?
      "contenttypes": [
        (contenttype)#
      ]?
    }#
  ],?
  "entitytagging": (entitytagging),
  "pagedretrieval": (pagedretrieval)
}
```

Where:

- ***(self)***, ***(enumeration)***, ***(creation)***, ***(entitytagging)***, and ***(pagedretrieval)*** are the values of the like-named attributes of the represented ServerEntryPoint payload element, using the representation defined in 6.2.6.
- ***(static-method-name/uri)***, ***(name)***, ***(protocolversion)***, and ***(contenttype)*** are single entries in the respective array attributes, using the representations defined in 6.2.6.
  If one of these arrays in the JSON representation has no entries, the corresponding JSON object member should not be present (but may be present with a value of an empty JSON array).

Example:

```
{
  "kind": "serverentrypoint",
  "self": "/cimrs",
  "namespaces": [
    { "name": "interop",
      "enumeration": "/cimrs/interop/enum",
      "creation": "/cimrs/interop/create",
      "staticmethods": [ MyStatic": "/cimrs/interop/mystatic" ],
      "protocolversions": [ "1.0.0", "1.0.1" ],
      "contenttypes": [
        "application/json;version=1.0.0",
        "application/json;version=1.0.1" ]
    },
    { "name": "root/cimv2",
      "enumeration": "/cimrs/root%2Fcimv2/enum",
      "creation": "/cimrs/root%2Fcimv2/create",
      "staticmethods": [ MyStatic": "/cimrs/root%2Fcimv2/mystatic" ],
      "protocolversions": [ "1.0.0", "1.0.1" ],
      "contenttypes": [
        "application/json;version=1.0.0",
        "application/json;version=1.0.1" ]
    }
  ],
```

```
  "entitytagging": true,
  "pagedretrieval": true
}
```

6.3.3 ListenerEntryPoint payload element

ListenerEntryPoint payload elements (see DSP0210) shall be represented using the following JSON structure:

```
{
  "kind": "listenerentrypoint"
  "self": (self),
  "destinations": [
    (destination)#
  ],?
  "protocolversions": [
    (protocolversion)#
  ],?
  "contenttypes": [
    (contenttype)#
  ]?
}
```

Where:

- ***(self)*** is the value of the like-named attribute of the represented ListenerEntryPoint payload element, using the representation defined in 6.2.6.
- ***(destination)***, ***(protocolversion)***, and ***(contenttype)*** are single entries in the array attributes *destinations*, *protocolversions*, and *contenttypes*, respectively, of the represented ListenerEntryPoint payload element, using the representations defined in 6.2.6.
  If one of these arrays in the JSON representation has no entries, the corresponding JSON object member should not be present (but may be present with a value of an empty JSON array).

Example:

```
{
  "kind": "listenerentrypoint",
  "self": "/cimrs",
  "destinations": [ "/cimrs/dest1", "/cimrs/dest2" ],
  "protocolversions": [ "1.0.0" ],
  "contenttypes": [
    "application/json;version=1.0.0" ]
}
```

6.3.4 Instance payload element

Instance payload elements (see DSP0210) shall be represented using the following JSON structure:

```
{
  "kind": "instance",
  "self": (self),
  "class": (class),
```

```
  "properties": {
    (property-name): (property-value)#
  },?
  "methods": {
    (method-name): (method-uri)#
  }?
}
```

Where:

- ***(self)*** and ***(class)*** are the values of the like-named attributes of the represented Instance payload element, using the representation defined in 6.2.6.
- Each member of *properties* represents an entry in the *properties* array attribute of the represented Instance payload element; that is, a property of the represented instance. ***(property-name)*** is the property name; ***(property-value)*** is the property value represented as defined in 6.2.7.
  If the *properties* array of the represented Instance payload element has no entries, the *properties* JSON object member should not be present (but may be present with a value of an empty JSON object).
- Each member of *methods* represents an entry in the *methods* array attribute of the represented Instance payload element; that is, a method invocation link of the represented instance. ***(method-name)*** is the method name; ***(method-uri)*** is the resource identifier represented as defined in 6.2.6 for abstract datatype URI.
  If the *methods* array of the represented Instance payload element has no entries, the *methods* JSON object member should not be present (but may be present with a value of an empty JSON object).

Example:

```
{
  "kind": "instance",
  "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0",
  "class": "CIM_RegisteredProfile",
  "properties": {
    "InstanceID": "DMTF:Fan:1.1.0",
    "RegisteredName": "Fan",
    "RegisteredOrganization": 2,
    "RegisteredVersion": "1.1.0" }
  "methods": {
    "GetCentralInstanceNames": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFa
n%3A1.1.0/GetCentralInstanceNames" }
}
```

6.3.5 InstanceCollection payload element

InstanceCollection payload elements (see DSP0210) shall be represented using the following JSON structure:

```
{
  "kind": "instancecollection",
  "self": (self),
  "next": (next),?
  "class": (class),
```

```
  "instances": [
    (instance)#
  ]?
}
```

Where:

- ***(self)***, ***(next)*** and ***(class)*** are the values of the like-named attributes of the represented InstanceCollection payload element, using the representation defined in 6.2.6.
- Each array entry of *instances* represents an entry in the *instances* array attribute of the represented InstanceCollection payload element; that is, an instance of the represented instance collection. ***(instance)*** is a representation of the instance as defined in 6.2.6 for abstract datatype Instance.
  If the array attribute *instances* of the represented InstanceCollection payload element has no entries, the *instances* JSON object member should not be present (but may be present with a value of an empty JSON array).

Example for an entire collection (that is, not in paged mode):

```
{
  "kind": "instancecollection",
  "self": "/cimrs/enum?namespace=root%2Fcimv2&class=CIM_System",
  "class": "CIM_System",
  "instances": [
    {
      "kind": "instance",
      "self": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1",
      "class": "CIM_ComputerSystem",
      "properties": {
        "InstanceID": "sys:1",
        "ElementName": "System #1" }
      "methods": {
        "RequestStateChange": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1/Request
StateChange" }
    },{
      "kind": "instance",
      "self": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys1",
      "class": "CIM_ComputerSystem",
      "properties": {
        "InstanceID": "sys:2",
        "ElementName": null }
      "methods": {
        "RequestStateChange": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:2/Request
StateChange" }
    } ]
}
```

NOTE: This example assumes that CIM_ComputerSystem is a subclass of CIM_System.

6.3.6 MethodInvocationRequest payload element

MethodInvocationRequest payload elements (see DSP0210) shall be represented using the following JSON structure:

```
{
  "kind": "methodrequest",
  "self": (self),
  "method": (method),
  "parameters": {
    (parameter-name): (parameter-value)#
  }
}
```

Where:

- ***(self)*** and ***(method)*** are the values of the like-named attributes of the represented MethodInvocationRequest payload element, using the representation defined in 6.2.6.
- Each member of *parameters* represents an entry in the *parameters* array attribute of the represented MethodInvocationRequest payload element; that is, a parameter of the request. ***(parameter-name)*** is the parameter name; ***(parameter-value)*** is the parameter value represented as defined in 6.2.7.
  If the *parameters* array of the represented MethodInvocationRequest payload element has no entries, the *parameters* JSON object member should not be present (but may be present with a value of an empty JSON object).

Example:

```
{
  "kind": "methodrequest",
  "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentralI
nstanceNames",
  "method": "GetCentralInstanceNames",
  "parameters": {
    "MaxNumber": 100 }
}
```

6.3.7 MethodInvocationResponse payload element

MethodInvocationResponse payload elements (see DSP0210) shall be represented using the following JSON structure:

```
{
  "kind": "methodresponse",
  "self": (self),
  "method": (method),
  "returnvalue": (return-value),
  "parameters": {
    (parameter-name): (parameter-value)#
  }
}
```

Where:

- ***(self)*** and ***(method)*** are the values of the like-named attributes of the represented MethodInvocationResponse payload element, using the representation defined in 6.2.6.
- *returnvalue* represents the *returnvalue* attribute of the represented MethodInvocationResponse payload element; that is, the return value of the method. ***(return-value)*** is the return value represented as defined in 6.2.7.

- Each member of *parameters* represents an entry in the *parameters* array attribute of the represented MethodInvocationResponse payload element; that is, an output parameter of the method. ***(parameter-name)*** is the parameter name; ***(parameter-value)*** is the parameter value represented as defined in 6.2.7.
  If the *parameters* array of the represented MethodInvocationResponse payload element has no entries, the *parameters* JSON object member should not be present (but may be present with a value of an empty JSON object).

Example:

```
{
  "kind": "methodresponse",
  "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentralI
nstanceNames",
  "method": "GetCentralInstanceNames",
  "returnvalue": 0,
  "parameters": {
    "CentralInstanceNames": [
        "/cimrs/root%2Fcimv2/CIM_Fan/fan:1",
        "/cimrs/root%2Fcimv2/CIM_Fan/fan:2" ]
  }
}
```

6.3.8 IndicationDeliveryRequest payload element

IndicationDeliveryRequest payload elements (see DSP0210) shall be represented using the following JSON structure:

```
{
  "kind": "indicationdeliveryrequest",
  "self": (self),
  "indication": (indication-instance)
}
```

Where:

- ***(self)*** and ***(indication)*** are the values of the like-named attributes of the represented IndicationDeliveryRequest payload element, using the representations defined in 6.2.6.
- ***(indication-instance)*** is the value of the attribute *indication* of the represented IndicationDeliveryRequest payload element, using the representation for abstract type Instance defined in 6.2.6.

Example:

```
{
  "kind": "indicationdeliveryrequest",
  "self": "/cimrs/dest1",
  "indication": {
    "kind": "instance",
    "class": "CIM_AlertIndication",
    "properties": {
      "AlertType": 4,
      "PerceivedSeverity": 6,
      "ProbableCause": 20,
```

```
      "Message": "ACME0007: Flood detected, height=3m.",
      "MessageArguments": [ "3" ],
      "MessageID": "ACME0007",
      "OwningEntity": "ACME" }
  }
}
```

6.3.9 ErrorResponse payload element

ErrorResponse payload elements (see DSP0210) shall be represented using the following JSON structure:

```
{
  "kind": "errorresponse",
  "self": (self),
  "httpmethod": (httpmethod),
  "statuscode": (statuscode),
  "statusdescription": (statusdescription),
  "errors": [
    (error-instance)#
  ]?
}
```

Where:

- ***(self)***, ***(httpmethod)***, ***(statuscode)***, and ***(statusdescription)*** are the values of the like-named attributes of the represented ErrorResponse payload element, using the representation defined in 6.2.6.
- Each array entry of *errors* represents an entry in the *errors* array attribute of the represented ErrorResponse payload element; that is, an instance of class CIM_Error. ***(error-instance)*** is a representation of the instance as defined in 6.2.6 for abstract datatype Instance.
  If the array attribute *errors* of the represented ErrorResponse payload element has no entries, the *errors* JSON object member should not be present (but may be present with a value of an empty JSON array).

Example:

```
{
  "kind": "errorresponse,
  "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0",
  "httpmethod": "GET",
  "statuscode": 12,
  "statusdescription": "ACME0008: Control program terminated with rc=42.",
  "errors": [
    {
      "kind": "instance",
      "class": "CIM_Error",
      "properties": {
        "ErrorType": 4,
        "PerceivedSeverity": 5,
        "ProbableCause": 48,
        "Message": "ACME0008: Control program terminated with rc=42.",
        "MessageArguments": [ "42" ],
```

```
        "MessageID": "ACME0008",
        "OwningEntity": "ACME" }
    } ]
}
```

ANNEX A
(informative)

Change log

Version	Date	Description
1.0.0a	2012-08-28	Released as a Work in Progress

Bibliography

This bibliography contains a list of non-normative references for this document.

DMTF DSP2032, *CIM-RS White Paper 1.0*,
http://www.dmtf.org/standards/published_documents/DSP2032_1.0.pdf

IANA MIME Media Types,
http://www.iana.org/assignments/media-types/

J. Holzer, *RESTful Web Services and JSON for WBEM Operations*, Master thesis, University of Applied Sciences, Konstanz, Germany, June 2009,
http://mond.htwg-konstanz.de/Abschlussarbeiten/Details.aspx?id=1120