



1  
2  
3  
4  
5

**Document Number: DSP0211**

**Date: 2012-08-28**

**Version: 1.0.0a**

## 6 **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>

7 **Document Type: Specification**

8 **Document Status: Work in Progress**

9 **Document Language: en-US**

10

11 Copyright Notice

12 Copyright © 2010-2012 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

13 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems  
14 management and interoperability. Members and non-members may reproduce DMTF specifications and  
15 documents, provided that correct attribution is given. As DMTF specifications may be revised from time to  
16 time, the particular version and release date should always be noted.

17 Implementation of certain elements of this standard or proposed standard may be subject to third party  
18 patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations  
19 to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose,  
20 or identify any or all such third party patent right, owners or claimants, nor for any incomplete or  
21 inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to  
22 any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize,  
23 disclose, or identify any such third party patent rights, or for such party's reliance on the standard or  
24 incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any  
25 party implementing such standard, whether such implementation is foreseeable or not, nor to any patent  
26 owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is  
27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  
28 implementing the standard from any and all claims of infringement by a patent owner for such  
29 implementations.

30 For information about patents held by third-parties which have notified the DMTF that, in their opinion,  
31 such patent may relate to or impact implementations of DMTF standards, visit  
32 <http://www.dmtf.org/about/policies/disclosures.php>.

33 **CONTENTS**

34 Foreword ..... 4

35 Acknowledgments ..... 4

36 Introduction..... 5

37 Document conventions..... 5

38 1 Scope ..... 6

39 2 Normative references ..... 6

40 3 Terms and definitions ..... 7

41 4 Symbols and abbreviated terms..... 8

42 5 Conformance..... 9

43 6 CIM-RS payload representation in JSON ..... 10

44 6.1 Overview ..... 10

45 6.2 General requirements ..... 10

46 6.2.1 Conformance to the JSON grammar ..... 10

47 6.2.2 Whitespace ..... 10

48 6.2.3 Character repertoire, representation, encoding and escaping ..... 10

49 6.2.4 Version of the payload representation in JSON ..... 11

50 6.2.5 Internet media type ..... 11

51 6.2.6 Representation of CIM-RS abstract datatypes ..... 12

52 6.2.7 Representation of CIM element values..... 12

53 6.2.8 Representation of CIM real32 and real64 datatypes ..... 14

54 6.2.9 Representation of CIM reference datatypes ..... 14

55 6.2.10 Representation of CIM Null values ..... 14

56 6.2.11 Representation of method invocation links ..... 15

57 6.3 Representation of protocol payload elements ..... 15

58 6.3.1 Format of payload element descriptions..... 15

59 6.3.2 ServerEntryPoint payload element ..... 16

60 6.3.3 ListenerEntryPoint payload element ..... 18

61 6.3.4 Instance payload element..... 18

62 6.3.5 InstanceCollection payload element ..... 19

63 6.3.6 MethodInvocationRequest payload element ..... 20

64 6.3.7 MethodInvocationResponse payload element..... 21

65 6.3.8 IndicationDeliveryRequest payload element ..... 22

66 6.3.9 ErrorResponse payload element ..... 23

67 ANNEX A (informative) Change log ..... 25

68 Bibliography ..... 26

69

70 **Tables**

71 Table 1 – Representation of CIM-RS abstract datatypes in JSON..... 12

72 Table 2 – Representation of CIM datatypes in JSON ..... 12

73 Table 3 – CIM-RS payload elements ..... 15

74

75

## Foreword

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

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

## 80 Acknowledgments

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

- 82 • Cornelia Davis, EMC
- 83 • George Ericson, EMC
- 84 • Johannes Holzer, IBM
- 85 • Robert Kieninger, IBM
- 86 • Wojtek Kozaczynski, Microsoft
- 87 • Larry Lamers, VMware
- 88 • Andreas Maier, IBM (editor)
- 89 • Bob Tillman, EMC
- 90 • Marvin Waschke, CA Technologies

91

## Introduction

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

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

## 96 Document conventions

97 Typographical conventions

98 The following typographical conventions are used in this document:

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

101 ABNF usage conventions

102 Format definitions in this document are specified using ABNF (see [RFC5234](#)), with the following  
103 deviations:

- 104 • Literal strings are to be interpreted as case-sensitive UCS characters, as opposed to the  
105 definition in [RFC5234](#) that interprets literal strings as case-insensitive US-ASCII characters.

106

# CIM-RS Payload Representation in JSON

## 107 1 Scope

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

111 Specifically, it describes how the abstract payload elements defined in [DSP0210](#) are represented in  
112 JSON and how a JSON representation of these payload elements is identified using an Internet media  
113 type.

114 Background information for CIM-RS is described in a white paper, [DSP2032](#).

## 115 2 Normative references

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

120 ANSI/IEEE 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*, August 1985,  
121 [http://ieeexplore.ieee.org/xpl/freeabs\\_all.jsp?arnumber=30711](http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=30711)

122 DMTF DSP0004, *CIM Infrastructure Specification 2.7*,  
123 [http://www.dmtf.org/standards/published\\_documents/DSP0004\\_2.7.pdf](http://www.dmtf.org/standards/published_documents/DSP0004_2.7.pdf)

124 DMTF DSP0210, *CIM-RS Protocol 1.0*,  
125 [http://www.dmtf.org/standards/published\\_documents/DSP0210\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0210_1.0.pdf)

126 DMTF DSP0223, *Generic Operations 1.0*,  
127 [http://www.dmtf.org/standards/published\\_documents/DSP0223\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf)

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

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

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

135 ISO/IEC 10646:2003, *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*,  
136 [http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921\\_ISO\\_IEC\\_10646\\_2003\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921_ISO_IEC_10646_2003(E).zip)

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

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

### 143 3 Terms and definitions

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

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

152 The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as  
153 described in [ISO/IEC Directives, Part 2](#), Clause 5.

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

157 The terms defined in [DSP0004](#), [DSP0223](#), and [DSP0210](#) apply to this document. Specifically, this  
158 document uses the terms "namespace", "qualifier", "qualifier type", "class", "creation class", "ordinary  
159 class", "association", "indication", "instance", "property", "ordinary property", "reference", "method",  
160 "parameter", and "return value" defined in [DSP0004](#).

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

#### 163 3.1

##### 164 CIM-RS payload element

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

#### 168 3.2

##### 169 CIM-RS payload representation

170 an encoding format that defines how the abstract payload elements defined in [DSP0210](#) are encoded in  
171 the entity body of the HTTP messages used by the CIM-RS protocol. This includes resource  
172 representations.

#### 173 3.3

##### 174 CIM-RS payload representation specification

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

#### 176 3.4

##### 177 CIM-RS protocol

178 the RESTful protocol defined in [DSP0210](#), for which this document describes a payload representation in  
179 JSON.

#### 180 3.1

##### 181 CIM-RS resource

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

#### 185 3.2

**186 CIM-RS resource identifier**

187 a URI that is a reference to a CIM-RS resource in a WBEM server or WBEM listener, as defined in  
188 [DSP0210](#). Also called "resource identifier" in this document.

**189 3.3****190 Internet media type**

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

**194 3.4****195 Normalization Form C**

196 a normalization form for UCS characters that avoids the use of combining marks where possible and that  
197 allows comparing UCS character strings on a per-code-point basis. It is defined in [The Unicode Standard,](#)  
198 [Annex #15](#).

**199 3.1****200 resource representation**

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

**205 3.2****206 UCS character**

207 a character from the Universal Character Set defined in [ISO/IEC 10646:2003](#). See [DSP0004](#) for the  
208 usage of UCS characters in CIM strings. An alternative term is "Unicode character".

**209 3.3****210 UCS code position**

211 a numeric identification for a UCS character in the range of 0x0 to 0x10FFFF, as defined in [ISO/IEC](#)  
212 [10646:2003](#).

**213 3.4****214 WBEM client**

215 the client role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see [DSP0210](#).

**216 3.5****217 WBEM listener**

218 the event listener role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see  
219 [DSP0210](#).

**220 3.6****221 WBEM server**

222 the server role in the CIM-RS protocol and in other WBEM protocols. For a full definition, see [DSP0210](#).

**223 4 Symbols and abbreviated terms**

224 The abbreviations defined in [DSP0004](#), [DSP0223](#), and [DSP0210](#) apply to this document. The following  
225 additional abbreviations are used in this document.

**226 4.1**



- 227 **ABNF**  
228 Augmented Backus-Naur Form, as defined in [RFC5234](#).
- 229 **4.2**  
230 **CIM**  
231 Common Information Model, as defined by DMTF.
- 232 **4.3**  
233 **CIM-RS**  
234 **CIM RESTful Services**  
235 The RESTful protocol for CIM defined in this document [and related documents](#).
- 236 **4.4**  
237 **ECMAScript**  
238 a scripting language that is the standard version of what was called JavaScript. It is defined in [ECMA-262](#).  
239 [262](#).
- 240 **4.5**  
241 **IANA**  
242 Internet Assigned Numbers Authority; see <http://www.iana.org>.
- 243 **4.6**  
244 **JSON**  
245 JavaScript Object Notation, as defined in [ECMA-262](#).
- 246 **4.7**  
247 **REST**  
248 Representational State Transfer, as originally and informally described in [Architectural Styles and the Design of Network-based Software Architectures](#).  
249 [Architectural Styles and the Design of Network-based Software Architectures](#).
- 250 **4.8**  
251 **UCS**  
252 Universal Character Set, as defined in [ISO/IEC 10646:2003](#).
- 253 **4.9**  
254 **URI**  
255 Uniform Resource Identifier, as defined in [RFC3986](#).
- 256 **4.10**  
257 **UTF-8**  
258 UCS Transformation Format 8, as defined in [ISO/IEC 10646:2003](#).
- 259 **4.11**  
260 **WBEM**  
261 Web Based Enterprise Management, as defined by DMTF.

## 262 **5 Conformance**

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

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

## 267 **6 CIM-RS payload representation in JSON**

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

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

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

### 278 **6.1 Overview**

279 This subclause describes informally and at a high level how the CIM-RS payload elements defined in  
280 [DSP0210](#) are represented in JSON.

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

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

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

### 293 **6.2 General requirements**

#### 294 **6.2.1 Conformance to the JSON grammar**

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

#### 297 **6.2.2 Whitespace**

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

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

#### 303 **6.2.3 Character repertoire, representation, encoding and escaping**

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

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

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

- 312 1) The character repertoire of *SourceCharacter* shall be that defined for values of the CIM string  
 313 type (defined in [DSP0004](#)). Note that this character repertoire is larger than the character  
 314 repertoire defined by [ECMA-262](#).
- 315 2) *SourceCharacter* shall be represented in Normalization Form C.
- 316 3) *SourceCharacter* shall be encoded in UTF-8. As a consequence, the entire payload element will  
 317 be encoded in UTF-8, and that character encoding is therefore not being indicated in the CIM-  
 318 RS payload or in any HTTP header fields.

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

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

```
326 UnicodeEscapeSequence ::
327     u HexDigit HexDigit HexDigit HexDigit
328     u HexDigit HexDigit HexDigit HexDigit HexDigit
329     u HexDigit HexDigit HexDigit HexDigit HexDigit HexDigit
```

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

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

## 335 6.2.4 Version of the payload representation in JSON

336 [DSP0210](#) requires that CIM-RS payload representation specifications define a version for the payload  
 337 representations they define.

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

## 340 6.2.5 Internet media type

341 [DSP0210](#) requires that CIM-RS payload representation specifications define a unique Internet media type  
 342 that identifies the representation.

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

346 Note that [DSP0210](#) defines requirements for specifying parameters on media types that identify the  
 347 representation of CIM-RS payload elements. One example for such a parameter is "version", specifying  
 348 the version of the payload representation.

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

350 `application/json; version=1.0.0`

### 351 6.2.6 Representation of CIM-RS abstract datatypes

352 This subclause defines how values of the abstract datatypes used in [DSP0210](#) for the definition of the  
353 attributes of the abstract payload elements are represented in JSON.

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

355 **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 <a href="#">DSP0210</a> ).
Instance	object	See 6.3.4

### 356 6.2.7 Representation of CIM element values

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

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

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

366 **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 <i>class</i> 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 <i>class</i> 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 <a href="#">DSP0004</a>
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

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

```

368 . . .
369 "ABoolean": true,
370 "AString": "some text",
371 "AChar16": "Z",
372 "AnOctetstringViaString": "0x00000007616263",
373 "AnOctetstringViaUint8Array": [ 0, 0, 0, 7, 0x61, 0x62, 0x63 ],
374 "AnEmbeddedInstance": {
375   "kind": "instance",
376   "class": "CIM_ComputerSystem",
377   "properties": {
378     "InstanceID": "sys:1",
379     "ElementName": "system #1" }
380 },
381 "AnEmbeddedObjectThatIsAnInstance": {
382   "kind": "instance",
383   "class": "CIM_ComputerSystem",
384   "properties": {
385     "InstanceID": "sys:1",
386     "ElementName": "system #1" }
387 },
388 "ADatetime": "20120213175830.123456+060",
389 "AUint16": 20000,
390 "ASint16": -16000,
391 "AReal32": 3.1415927,
392 "ARef": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1",
393 "ABooleanArray": [ true, false, true ],

```

```
394     "AStringArray": [ "some text", null, "more text\n" ],
395     "AReal64Array": [ 1E-42, NaN, "-Infinity" ],
396     . . .
```

## 397 6.2.8 Representation of CIM real32 and real64 datatypes

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

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

402 positive infinity: "Infinity"

403 negative infinity: "-Infinity"

404 not-a-number: "NaN"

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

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

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

417 Examples:

```
418     . . .
419     "Throughput": 3.45E3,
420     "ErrorRate": "NaN",
421     . . .
```

## 422 6.2.9 Representation of CIM reference datatypes

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

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

427 Example for a reference property named System that is declared as type "CIM\_ComputerSystem ref":

```
428     . . .
429     "System": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1"
430     . . .
```

## 431 6.2.10 Representation of CIM Null values

432 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 {
```



```

457     "kind": "instance",
458     "self": (value),
459     "class": (value),
460     "properties": {
461         (value): (value)#
462     },?
463     "methods": {
464         (value): (value)#
465     }?
466 }

```

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

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

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

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

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

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

```

480 {
481     "kind": "instance",
482     "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0",
483     "class": "CIM_RegisteredProfile",
484     "properties": {
485         "InstanceID": "DMTF:Fan:1.1.0",
486         "RegisteredName": "Fan",,
487         "RegisteredOrganization": 2,
488         "RegisteredVersion": "1.1.0"
489     }
490 }

```

### 491 6.3.2 ServerEntryPoint payload element

492 ServerEntryPoint payload elements (see [DSP0210](#)) shall be represented using the following JSON  
 493 structure:

```

494 {
495     "kind": "serverentrypoint"
496     "self": (self),
497     "namespaces": [
498         { "name": (name),
499           "enumeration": (enumeration),
500           "creation": (creation),

```



```

501     "staticmethods": [
502         (static-method-name): (static-method-uri)#
503     ],?
504     "protocolversions": [
505         (protocolversion)#
506     ],?
507     "contenttypes": [
508         (contenttype)#
509     ]?
510     }#
511 ],?
512 "entitytagging": (entitytagging),
513 "pagedretrieval": (pagedretrieval)
514 }

```

515 Where:

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

524 Example:

```

525 {
526     "kind": "serverentrypoint",
527     "self": "/cimrs",
528     "namespaces": [
529         { "name": "interop",
530           "enumeration": "/cimrs/interop/enum",
531           "creation": "/cimrs/interop/create",
532           "staticmethods": [ MyStatic: "/cimrs/interop/mystatic" ],
533           "protocolversions": [ "1.0.0", "1.0.1" ],
534           "contenttypes": [
535             "application/json;version=1.0.0",
536             "application/json;version=1.0.1" ]
537         },
538         { "name": "root/cimv2",
539           "enumeration": "/cimrs/root%2Fcimv2/enum",
540           "creation": "/cimrs/root%2Fcimv2/create",
541           "staticmethods": [ MyStatic: "/cimrs/root%2Fcimv2/mystatic" ],
542           "protocolversions": [ "1.0.0", "1.0.1" ],
543           "contenttypes": [
544             "application/json;version=1.0.0",
545             "application/json;version=1.0.1" ]
546         }
547     ],

```

```

548     "entitytagging": true,
549     "pagedretrieval": true
550 }

```

### 551 6.3.3 ListenerEntryPoint payload element

552 ListenerEntryPoint payload elements (see [DSP0210](#)) shall be represented using the following JSON  
553 structure:

```

554 {
555     "kind": "listenerentrypoint"
556     "self": (self),
557     "destinations": [
558         (destination)#
559     ],?
560     "protocolversions": [
561         (protocolversion)#
562     ],?
563     "contenttypes": [
564         (contenttype)#
565     ]?
566 }

```

567 Where:

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

576 Example:

```

577 {
578     "kind": "listenerentrypoint",
579     "self": "/cimrs",
580     "destinations": [ "/cimrs/dest1", "/cimrs/dest2" ],
581     "protocolversions": [ "1.0.0" ],
582     "contenttypes": [
583         "application/json;version=1.0.0" ]
584 }

```

### 585 6.3.4 Instance payload element

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

```

587 {
588     "kind": "instance",
589     "self": (self),
590     "class": (class),

```

```

591  "properties": {
592      (property-name): (property-value)#
593  },?
594  "methods": {
595      (method-name): (method-uri)#
596  }?
597  }

```

598 Where:

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

615 Example:

```

616  {
617      "kind": "instance",
618      "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0",
619      "class": "CIM_RegisteredProfile",
620      "properties": {
621          "InstanceID": "DMTF:Fan:1.1.0",
622          "RegisteredName": "Fan",
623          "RegisteredOrganization": 2,
624          "RegisteredVersion": "1.1.0" }
625      "methods": {
626          "GetCentralInstanceNames": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFa
627 n%3A1.1.0/GetCentralInstanceNames" }
628  }

```

### 629 6.3.5 InstanceCollection payload element

630 InstanceCollection payload elements (see [DSP0210](#)) shall be represented using the following JSON  
631 structure:

```

632  {
633      "kind": "instancecollection",
634      "self": (self),
635      "next": (next),?
636      "class": (class),

```

```

637 "instances": [
638     (instance)#
639 ]?
640 }

```

641 Where:

- 642 • (**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.
- 643
- 644 • 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.
- 645
- 646 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).
- 647
- 648
- 649
- 650

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

```

652 {
653     "kind": "instancecollection",
654     "self": "/cimrs/enum?namespace=root%2Fcimv2&class=CIM_System",
655     "class": "CIM_System",
656     "instances": [
657         {
658             "kind": "instance",
659             "self": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1",
660             "class": "CIM_ComputerSystem",
661             "properties": {
662                 "InstanceID": "sys:1",
663                 "ElementName": "System #1" }
664             "methods": {
665                 "RequestStateChange": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1/Request
666 StateChange" }
667         }, {
668             "kind": "instance",
669             "self": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys1",
670             "class": "CIM_ComputerSystem",
671             "properties": {
672                 "InstanceID": "sys:2",
673                 "ElementName": null }
674             "methods": {
675                 "RequestStateChange": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:2/Request
676 StateChange" }
677         } ]
678     }

```

679 NOTE: This example assumes that CIM\_ComputerSystem is a subclass of CIM\_System.

### 680 6.3.6 MethodInvocationRequest payload element

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

```

683 {
684   "kind": "methodrequest",
685   "self": (self),
686   "method": (method),
687   "parameters": {
688     (parameter-name): (parameter-value)#
689   }
690 }

```

691 Where:

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

701 Example:

```

702 {
703   "kind": "methodrequest",
704   "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentralI
705 nstanceNames",
706   "method": "GetCentralInstanceNames",
707   "parameters": {
708     "MaxNumber": 100 }
709 }

```

### 710 6.3.7 MethodInvocationResponse payload element

711 MethodInvocationResponse payload elements (see [DSP0210](#)) shall be represented using the following  
712 JSON structure:

```

713 {
714   "kind": "methodresponse",
715   "self": (self),
716   "method": (method),
717   "returnvalue": (return-value),
718   "parameters": {
719     (parameter-name): (parameter-value)#
720   }
721 }

```

722 Where:

- 723 • (*self*) and (*method*) are the values of the like-named attributes of the represented  
724 MethodInvocationResponse payload element, using the representation defined in 6.2.6.
- 725 • *returnvalue* represents the *returnvalue* attribute of the represented MethodInvocationResponse  
726 payload element; that is, the return value of the method. (*return-value*) is the return value  
727 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:

```

736 {
737   "kind": "methodresponse",
738   "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentralI
739 nstanceNames",
740   "method": "GetCentralInstanceNames",
741   "returnvalue": 0,
742   "parameters": {
743     "CentralInstanceNames": [
744       "/cimrs/root%2Fcimv2/CIM_Fan/fan:1",
745       "/cimrs/root%2Fcimv2/CIM_Fan/fan:2" ]
746   }
747 }
```

### 748 6.3.8 IndicationDeliveryRequest payload element

749 IndicationDeliveryRequest payload elements (see [DSP0210](#)) shall be represented using the following  
750 JSON structure:

```

751 {
752   "kind": "indicationdeliveryrequest",
753   "self": (self),
754   "indication": (indication-instance)
755 }
```

756 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.

762 Example:

```

763 {
764   "kind": "indicationdeliveryrequest",
765   "self": "/cimrs/dest1",
766   "indication": {
767     "kind": "instance",
768     "class": "CIM_AlertIndication",
769     "properties": {
770       "AlertType": 4,
771       "PerceivedSeverity": 6,
772       "ProbableCause": 20,
```

```

773     "Message": "ACME0007: Flood detected, height=3m.",
774     "MessageArguments": [ "3" ],
775     "MessageID": "ACME0007",
776     "OwningEntity": "ACME" }
777 }
778 }

```

### 779 6.3.9 ErrorResponse payload element

780 ErrorResponse payload elements (see [DSP0210](#)) shall be represented using the following JSON  
781 structure:

```

782 {
783   "kind": "errorresponse",
784   "self": (self),
785   "httpmethod": (httpmethod),
786   "statusCode": (statusCode),
787   "statusdescription": (statusdescription),
788   "errors": [
789     (error-instance)#
790   ]?
791 }

```

792 Where:

- 793 • (*self*), (*httpmethod*), (*statusCode*), and (*statusdescription*) are the values of the like-named  
794 attributes of the represented ErrorResponse payload element, using the representation defined  
795 in 6.2.6.
- 796 • Each array entry of *errors* represents an entry in the *errors* array attribute of the represented  
797 ErrorResponse payload element; that is, an instance of class CIM\_Error. (*error-instance*) is a  
798 representation of the instance as defined in 6.2.6 for abstract datatype Instance.  
799 If the array attribute *errors* of the represented ErrorResponse payload element has no entries,  
800 the *errors* JSON object member should not be present (but may be present with a value of an  
801 empty JSON array).

802 Example:

```

803 {
804   "kind": "errorresponse",
805   "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0",
806   "httpmethod": "GET",
807   "statusCode": 12,
808   "statusdescription": "ACME0008: Control program terminated with rc=42.",
809   "errors": [
810     {
811       "kind": "instance",
812       "class": "CIM_Error",
813       "properties": {
814         "ErrorType": 4,
815         "PerceivedSeverity": 5,
816         "ProbableCause": 48,
817         "Message": "ACME0008: Control program terminated with rc=42.",
818         "MessageArguments": [ "42" ],

```

```
819     "MessageID": "ACME0008",  
820     "OwningEntity": "ACME" }  
821 } ]  
822 }
```



823  
824  
825  
826  
827

## **ANNEX A** (informative)

### **Change log**

<b>Version</b>	<b>Date</b>	<b>Description</b>
1.0.0a	2012-08-28	Released as a Work in Progress

828

## Bibliography

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

830 DMTF DSP2032, *CIM-RS White Paper 1.0*,

831 [http://www.dmtf.org/standards/published\\_documents/DSP2032\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP2032_1.0.pdf)

832 IANA MIME Media Types,

833 <http://www.iana.org/assignments/media-types/>

834 J. Holzer, *RESTful Web Services and JSON for WBEM Operations*, Master thesis, University of Applied

835 Sciences, Konstanz, Germany, June 2009,

836 <http://mond.htwg-konstanz.de/Abschlussarbeiten/Details.aspx?id=1120>