



1  
2  
3  
4

**Document Number: DSP0211**

**Date: 2014-02-11**

**Version: 1.0.1**

## 5 **CIM-RS Payload Representation in JSON**

6 **Document Type: Specification**  
7 **Document Status: DMTF Standard**  
8 **Document Language: en-US**

9 Copyright Notice

10 Copyright © 2010–2014 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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25 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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

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

31

# CONTENTS

32 Foreword ..... 4

33 Introduction..... 5

34 Document conventions..... 5

35 Typographical conventions ..... 5

36 ABNF usage conventions ..... 5

37 1 Scope ..... 7

38 2 Normative references ..... 7

39 3 Terms and definitions ..... 8

40 4 Symbols and abbreviated terms..... 9

41 5 Conformance..... 10

42 6 CIM-RS payload representation in JSON ..... 11

43 6.1 Overview ..... 11

44 6.2 General requirements ..... 11

45 6.2.1 Conformance to the JSON grammar ..... 11

46 6.2.2 Whitespace ..... 11

47 6.2.3 Character repertoire, representation, encoding and escaping ..... 12

48 6.2.4 Version of the payload representation in JSON ..... 12

49 6.2.5 Internet media type ..... 12

50 6.2.6 Representation of CIM-RS abstract datatypes ..... 13

51 6.2.7 Representation of CIM element values..... 13

52 6.2.8 Representation of CIM real32 and real64 datatypes ..... 15

53 6.2.9 Representation of CIM reference datatypes ..... 15

54 6.2.10 Representation of CIM Null values ..... 16

55 6.2.11 Representation of method invocation links ..... 16

56 6.3 Representation of protocol payload elements ..... 16

57 6.3.1 Format of payload element descriptions..... 17

58 6.3.2 ServerEntryPoint payload element ..... 18

59 6.3.3 ListenerEntryPoint payload element ..... 19

60 6.3.4 Instance payload element..... 20

61 6.3.5 InstanceCollection payload element ..... 21

62 6.3.6 ReferenceCollection payload element ..... 22

63 6.3.7 MethodRequest payload element ..... 23

64 6.3.8 MethodResponse payload element ..... 23

65 6.3.9 IndicationDeliveryRequest payload element ..... 24

66 6.3.10 ErrorResponse payload element ..... 25

67 ANNEX A (informative) Change log ..... 27

68

## 69 Tables

70 Table 1 – Representation of CIM-RS abstract datatypes in JSON..... 13

71 Table 2 – Representation of CIM datatypes in JSON ..... 13

72 Table 3 – CIM-RS payload elements ..... 16

73

74

## Foreword

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

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

## 79 Acknowledgments

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

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

90

## Introduction

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

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

### 95 Document conventions

#### 96 Typographical conventions

97 The following typographical conventions are used in this document:

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

#### 100 ABNF usage conventions

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

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



107

# CIM-RS Payload Representation in JSON

## 1 Scope

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

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

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

## 2 Normative references

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

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

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

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

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

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

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

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

136 ISO/IEC 10646:2003, *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*,  
137 [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)

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

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

### 144 3 Terms and definitions

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

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

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

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

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

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

#### 164 3.1

##### 165 CIM-RS payload element

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

#### 169 3.2

##### 170 CIM-RS payload representation

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

#### 174 3.3

##### 175 CIM-RS payload representation specification

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

#### 177 3.4

##### 178 CIM-RS protocol

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

#### 181 3.5

##### 182 CIM-RS resource

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



**186 3.6****187 CIM-RS resource identifier**

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

**190 3.7****191 Internet media type**

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

**195 3.8****196 Normalization Form C**

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

**200 3.9****201 resource representation**

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

**206 3.10****207 UCS character**

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

**210 3.11****211 UCS code position**

212 a numeric identification for a UCS character in the range of 0x0 to 0x10FFFF, as defined in [ISO/IEC](#)  
213 [10646:2003](#).

**214 3.12****215 WBEM client**

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

**217 3.13****218 WBEM listener**

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

**221 3.14****222 WBEM server**

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

**224 4 Symbols and abbreviated terms**

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

- 227 **4.1**  
228 **ABNF**  
229 Augmented Backus-Naur Form, as defined in [RFC5234](#).
- 230 **4.2**  
231 **CIM**  
232 Common Information Model, as defined by DMTF.
- 233 **4.3**  
234 **CIM-RS**  
235 **CIM RESTful Services**  
236 The RESTful protocol for CIM defined in this document and related documents.
- 237 **4.4**  
238 **ECMAScript**  
239 a scripting language that is the standard version of what was called JavaScript. It is defined in [ECMA-262](#).  
240
- 241 **4.5**  
242 **IANA**  
243 Internet Assigned Numbers Authority; see <http://www.iana.org>.
- 244 **4.6**  
245 **JSON**  
246 JavaScript Object Notation, as defined in [ECMA-262](#).
- 247 **4.7**  
248 **REST**  
249 Representational State Transfer, as originally and informally described in Architectural Styles and the  
250 Design of Network-based Software Architectures.
- 251 **4.8**  
252 **UCS**  
253 Universal Character Set, as defined in [ISO/IEC 10646:2003](#).
- 254 **4.9**  
255 **URI**  
256 Uniform Resource Identifier, as defined in [RFC3986](#).
- 257 **4.10**  
258 **UTF-8**  
259 UCS Transformation Format 8, as defined in [ISO/IEC 10646:2003](#).
- 260 **4.11**  
261 **WBEM**  
262 Web Based Enterprise Management, as defined by DMTF.

## 263 **5 Conformance**

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

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

## 268 **6 CIM-RS payload representation in JSON**

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

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

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

### 279 **6.1 Overview**

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

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

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

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

### 294 **6.2 General requirements**

#### 295 **6.2.1 Conformance to the JSON grammar**

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

#### 298 **6.2.2 Whitespace**

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

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

### 304 6.2.3 Character repertoire, representation, encoding and escaping

305 The JSON grammar defined in clause 15.12.1 of [ECMA-262](#) references the *SourceCharacter* symbol  
 306 defined in its clause 6 as the basis for the characters of its grammar be it for identifiers, delimiters or  
 307 values. The definition of the *SourceCharacter* symbol applies to the ECMAScript use of JSON and uses  
 308 the character repertoire of Unicode V3, requires a representation of UCS characters in Normalization  
 309 Form C, and effectively implies a requirement for an encoding in UTF-16 (or one of its little endian and big  
 310 endian derivatives).

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

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

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

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

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

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

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

### 336 6.2.4 Version of the payload representation in JSON

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

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

### 341 6.2.5 Internet media type

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

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

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

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

351 `application/json; version=1.0.0`

352 **6.2.6 Representation of CIM-RS abstract datatypes**

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

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

356 **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

357 **6.2.7 Representation of CIM element values**

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

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

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

367 **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)

CIM Datatype	JSON Datatype	Additional Rules
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)
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

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

```

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

```

```

392     "AReal32": 3.1415927,
393     "ARef": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1",
394     "ABooleanArray": [ true, false, true ],
395     "AStringArray": [ "some text", null, "more text\n" ],
396     "AReal64Array": [ 1E-42, NaN, "-Infinity" ],
397     . . .

```

## 398 6.2.8 Representation of CIM real32 and real64 datatypes

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

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

403           positive infinity: "Infinity"

404           negative infinity: "-Infinity"

405           not-a-number: "NaN"

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

409 NOTE The strings used for representing the IEEE special values are consistent with Python's serialization of float-  
410 typed values in JSON, and with Java's serialization of float-typed values as strings. These strings are not consistent  
411 with the representation of the special values in the XML datatypes 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 CIM  
413 real32 and real64 typed values is defined by the [IEEE 754](#) Single and Double formats, which have a basis of 2. The  
414 definition of a minimum precision for the significand guarantees that the value of CIM real types does not change  
415 when converting it back and forth between the (10-based) JSON representation and a (2-based) internal  
416 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

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

```
429     . . .
430     "System": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1"
431     . . .
```

432 **6.2.10 Representation of CIM Null values**

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

434 Note that the JSON literal null is case sensitive.

435 Example:

```
436     . . .
437     "ElementName": null,
438     "PossibleStates": [1, null, 3],
439     . . .
```

440 **6.2.11 Representation of method invocation links**

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

445 Example:

```
446     . . .
447     "RequestStateChange":
448         "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1/RequestStateChange"
449     . . .
```

450 **6.3 Representation of protocol payload elements**

451 This subclause defines how the CIM-RS payload elements defined in [DSP0210](#) are represented in JSON.

452 Table 3 lists the payload elements defined in [DSP0210](#).

453 **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
ReferenceCollection	representation of a set of CIM references in a reference collection	See 6.3.6
MethodRequest	the data used to request the invocation of a method	See 6.3.7



Payload Element	Meaning	Description
MethodResponse	the data used in the response of the invocation of a method	See 6.3.8
IndicationDeliveryRequest	the data used to request the delivery of an indication to a listener destination	See 6.3.9
ErrorResponse	the data used in an error response to any request	See 6.3.10

### 454 6.3.1 Format of payload element descriptions

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

```
457 {
458   "kind": "instance",
459   "self": (value),
460   "class": (value),
461   "properties": {
462     (value): (value)#
463   },?
464   "methods": {
465     (value): (value)#
466   }?
467 }
```

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

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

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

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

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

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

```

481 {
482   "kind": "instance",
483   "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0",
484   "class": "CIM_RegisteredProfile",
485   "properties": {
486     "InstanceID": "DMTF:Fan:1.1.0",
487     "RegisteredName": "Fan",,
488     "RegisteredOrganization": 2,
489     "RegisteredVersion": "1.1.0"
490   }
491 }

```

### 492 6.3.2 ServerEntryPoint payload element

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

```

495 {
496   "kind": "serverentrypoint"
497   "self": (self),
498   "namespaces": [
499     { "name": (name),
500       "enumeration": (enumeration),
501       "creation": (creation),
502       "staticmethods": [
503         (static-method-name): (static-method-uri)#
504       ],?
505       "protocolversions": [
506         (protocolversion)#
507       ],?
508       "contenttypes": [
509         (contenttype)#
510       ]?
511     }#
512   ],?
513   "entitytagging": (entitytagging),
514   "pagedretrieval": (pagedretrieval)
515 }

```

516 Where:

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

525 Example:

```

526 {
527   "kind": "serverentrypoint",
528   "self": "/cimrs",
529   "namespaces": [
530     { "name": "interop",
531       "enumeration": "/cimrs/interop/enum",
532       "creation": "/cimrs/interop/create",
533       "staticmethods": [ MyStatic: "/cimrs/interop/mystatic" ],
534       "protocolversions": [ "1.0.0", "1.0.1" ],
535       "contenttypes": [
536         "application/json;version=1.0.0",
537         "application/json;version=1.0.1" ]
538     },
539     { "name": "root/cimv2",
540       "enumeration": "/cimrs/root%2Fcimv2/enum",
541       "creation": "/cimrs/root%2Fcimv2/create",
542       "staticmethods": [ MyStatic: "/cimrs/root%2Fcimv2/mystatic" ],
543       "protocolversions": [ "1.0.0", "1.0.1" ],
544       "contenttypes": [
545         "application/json;version=1.0.0",
546         "application/json;version=1.0.1" ]
547     }
548   ],
549   "entitytagging": true,
550   "pagedretrieval": true
551 }

```

### 552 6.3.3 ListenerEntryPoint payload element

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

```

555 {
556   "kind": "listenerentrypoint"
557   "self": (self),
558   "destinations": [
559     (destination)#
560   ],?
561   "protocolversions": [
562     (protocolversion)#
563   ],?
564   "contenttypes": [
565     (contenttype)#
566   ]?
567 }

```

568 Where:

- 569 • (*self*) is the value of the like-named attribute of the represented ListenerEntryPoint payload  
570 element, using the representation defined in 6.2.6.

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

577 Example:

```
578 {
579   "kind": "listenerentrypoint",
580   "self": "/cimrs",
581   "destinations": [ "/cimrs/dest1", "/cimrs/dest2" ],
582   "protocolversions": [ "1.0.0" ],
583   "contenttypes": [
584     "application/json;version=1.0.0" ]
585 }
```

### 586 6.3.4 Instance payload element

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

```
588 {
589   "kind": "instance",
590   "self": (self),
591   "class": (class),
592   "properties": {
593     (property-name): (property-value)#
594   },?
595   "methods": {
596     (method-name): (method-uri)#
597   }?
598 }
```

599 Where:

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

616 Example:

```

617 {
618   "kind": "instance",
619   "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0",
620   "class": "CIM_RegisteredProfile",
621   "properties": {
622     "InstanceID": "DMTF:Fan:1.1.0",
623     "RegisteredName": "Fan",
624     "RegisteredOrganization": 2,
625     "RegisteredVersion": "1.1.0" }
626   "methods": {
627     "GetCentralInstanceNames": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFa
628     n%3A1.1.0/GetCentralInstanceNames" }
629 }

```

### 630 6.3.5 InstanceCollection payload element

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

```

633 {
634   "kind": "instancecollection",
635   "self": (self),
636   "next": (next), ?
637   "class": (class),
638   "instances": [
639     (instance)#
640   ]?
641 }

```

642 Where:

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

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

```

653 {
654   "kind": "instancecollection",
655   "self": "/cimrs/root%2Fcimv2/enum?$class=CIM_System",
656   "class": "CIM_System",
657   "instances": [
658     {
659       "kind": "instance",
660       "self": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1",

```

```

661     "class": "CIM_ComputerSystem",
662     "properties": {
663         "InstanceID": "sys:1",
664         "ElementName": "System #1" }
665     "methods": {
666         "RequestStateChange": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:1/Request
667 StateChange" }
668     },{
669     "kind": "instance",
670     "self": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys1",
671     "class": "CIM_ComputerSystem",
672     "properties": {
673         "InstanceID": "sys:2",
674         "ElementName": null }
675     "methods": {
676         "RequestStateChange": "/cimrs/root%2Fcimv2/CIM_ComputerSystem/sys:2/Request
677 StateChange" }
678     } ]
679 }

```

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

### 681 6.3.6 ReferenceCollection payload element

682 ReferenceCollection payload elements (see [DSP0210](#)) shall be represented using the following JSON  
683 structure:

```

684 {
685     "kind": "referencecollection",
686     "self": (self),
687     "next": (next), ?
688     "class": (class),
689     "references": [
690         (reference)#
691     ]?
692 }

```

693 Where:

- 694 • (**self**), (**next**) and (**class**) are the values of the like-named attributes of the represented  
695 ReferenceCollection payload element, using the representation defined in 6.2.6.
- 696 • Each array entry of *references* represents an entry in the *references* array attribute of the  
697 represented ReferenceCollection payload element; that is, a reference of the represented  
698 reference collection. (**reference**) is a representation of the reference as defined in 6.2.6 for  
699 abstract datatype URI.  
700 If the array attribute *references* of the represented ReferenceCollection payload element has no  
701 entries, the *references* JSON object member should not be present (but may be present with a  
702 value of an empty JSON array).

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

```

704 {
705     "kind": "referencecollection",
706     "self": "/cimrs/root%2Fcimv2/ACME_RegisteredProfile/DMTF%3AFan%3A1.0.0/refer/ACME

```

```

707     _ElementConformsToProfile.ManagedElement/part/2",
708     "class": "ACME_Fan",
709     "references": [
710         "/cimrs/root%2Fcimv2/ACME_Fan/fan11",
711         "/cimrs/root%2Fcimv2/ACME_Fan/fan12"
712     ]
713 }

```

### 714 6.3.7 MethodRequest payload element

715 MethodRequest payload elements (see [DSP0210](#)) shall be represented using the following JSON  
716 structure:

```

717 {
718     "kind": "methodrequest",
719     "self": (self),
720     "method": (method),
721     "parameters": {
722         (parameter-name): (parameter-value)#
723     }
724 }

```

725 Where:

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

735 Example:

```

736 {
737     "kind": "methodrequest",
738     "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentralI
739 nstanceNames",
740     "method": "GetCentralInstanceNames",
741     "parameters": {
742         "MaxNumber": 100 }
743 }

```

### 744 6.3.8 MethodResponse payload element

745 MethodResponse payload elements (see [DSP0210](#)) shall be represented using the following JSON  
746 structure:

```

747 {
748     "kind": "methodresponse",
749     "self": (self),
750     "method": (method),

```

```

751 "returnValue": (return-value),
752 "parameters": {
753   (parameter-name): (parameter-value)#
754 }
755 }

```

756 Where:

- 757 • **(self)** and **(method)** are the values of the like-named attributes of the represented  
758 MethodResponse payload element, using the representation defined in 6.2.6.
- 759 • *returnValue* represents the *returnValue* attribute of the represented MethodResponse payload  
760 element; that is, the return value of the method. **(return-value)** is the return value represented  
761 as defined in 6.2.7.
- 762 • Each member of *parameters* represents an entry in the *parameters* array attribute of the  
763 represented MethodResponse payload element; that is, an output parameter of the method.  
764 **(parameter-name)** is the parameter name; **(parameter-value)** is the parameter value  
765 represented as defined in 6.2.7.  
766 If the *parameters* array of the represented MethodResponse payload element has no entries,  
767 the *parameters* JSON object member should not be present (but may be present with a value of  
768 an empty JSON object).

769 Example:

```

770 {
771   "kind": "methodresponse",
772   "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0/GetCentralI
773 nstanceNames",
774   "method": "GetCentralInstanceNames",
775   "returnValue": 0,
776   "parameters": {
777     "CentralInstanceNames": [
778       "/cimrs/root%2Fcimv2/CIM_Fan/fan:1",
779       "/cimrs/root%2Fcimv2/CIM_Fan/fan:2" ]
780   }
781 }

```

### 782 6.3.9 IndicationDeliveryRequest payload element

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

```

785 {
786   "kind": "indicationdeliveryrequest",
787   "self": (self),
788   "indication": (indication-instance)
789 }

```

790 Where:

- 791 • **(self)** and **(indication)** are the values of the like-named attributes of the represented  
792 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.

796 Example:

```

797 {
798   "kind": "indicationdeliveryrequest",
799   "self": "/cimrs/dest1",
800   "indication": {
801     "kind": "instance",
802     "class": "CIM_AlertIndication",
803     "properties": {
804       "AlertType": 4,
805       "PerceivedSeverity": 6,
806       "ProbableCause": 20,
807       "Message": "ACME0007: Flood detected, height=3m.",
808       "MessageArguments": [ "3" ],
809       "MessageID": "ACME0007",
810       "OwningEntity": "ACME" }
811   }
812 }
```

### 813 6.3.10 ErrorResponse payload element

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

```

816 {
817   "kind": "errorresponse",
818   "self": (self),
819   "httpmethod": (httpmethod),
820   "statusCode": (statusCode),
821   "statusdescription": (statusdescription),
822   "errors": [
823     (error-instance)#
824   ]?
825 }
```

826 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).

836 Example:

```
837 {
838   "kind": "errorresponse",
839   "self": "/cimrs/root%2Fcimv2/CIM_RegisteredProfile/DMTF%3AFan%3A1.1.0",
840   "httpmethod": "GET",
841   "statuscode": 12,
842   "statusdescription": "ACME0008: Control program terminated with rc=42.",
843   "errors": [
844     {
845       "kind": "instance",
846       "class": "CIM_Error",
847       "properties": {
848         "ErrorType": 4,
849         "PerceivedSeverity": 5,
850         "ProbableCause": 48,
851         "Message": "ACME0008: Control program terminated with rc=42.",
852         "MessageArguments": [ "42" ],
853         "MessageID": "ACME0008",
854         "OwningEntity": "ACME" }
855     } ]
856 }
```

857  
858  
859  
860  
861

## ANNEX A (informative)

### Change log

Version	Date	Description
1.0.0	2013-01-24	
1.0.1	2014-02-11	Released as DMTF Standard, with the following changes: <ul style="list-style-type: none"><li>• Added missing description of ResourceCollection payload element (6.3.6)</li><li>• Fixed incorrect names of method invocation related payload elements (6.3.7 and 6.3.8)</li><li>• Fixed incorrect query parameters and syntax errors in examples</li></ul>

862

## Bibliography

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

864 DMTF DSP2032, *CIM-RS White Paper 1.0*,  
865 [http://www.dmtf.org/standards/published\\_documents/DSP2032\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP2032_1.0.pdf)

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

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