

1 Draft
2 OCCI-WG
3

Andy Edmons, ICCLab, ZHAW
Thijs Metsch, Intel
February 23, 2016

4 **Open Cloud Computing Interface – Text Rendering**

5 Status of this Document

6 This document provides information to the community regarding the specification of the Open Cloud Computing
7 Interface. Distribution is unlimited.

8 Copyright Notice

9 Copyright ©Open Grid Forum (2015-2016). All Rights Reserved.

10 Trademarks

11 OCCI is a trademark of the Open Grid Forum.

12 Abstract

13 This document, part of a document series produced by the OCCI working group within the Open Grid Forum
14 (OGF), provides a high-level definition of a Protocol and API. The document is based upon previously gathered
15 requirements and focuses on the scope of important capabilities required to support modern service offerings.

16	Contents	
17	1 Introduction	4
18	2 Notational Conventions	4
19	3 Text rendering	5
20	4 ABNF Definitions	5
21	4.1 Category ABNF	5
22	4.2 Link ABNF	5
23	4.3 Attribute ABNF	6
24	4.4 Location ABNF	6
25	5 Renderings	6
26	5.1 Entity Instance Rendering	6
27	5.1.1 Resource Instance Rendering	6
28	5.1.2 Link Instance Rendering	7
29	5.2 Category Instance Rendering	7
30	5.2.1 Kind Instance Rendering	7
31	5.2.2 Mixin Instance Rendering	7
32	5.2.3 Action Instance Rendering	7
33	5.3 Entity Collection Rendering	7
34	5.3.1 Resource Collection Rendering	7
35	5.3.2 Link Collection Rendering	7
36	5.4 Category Collection Rendering	8
37	5.4.1 Kind Collection Rendering	8
38	5.4.2 Mixin Collection Rendering	8
39	5.4.3 Action Collection Rendering	8
40	5.5 Attributes Rendering	8
41	5.5.1 Entity Instance Attribute Rendering Specifics	8
42	5.5.2 Mixin Instance Attribute Rendering Specifics	8
43	5.5.3 Attribute Description Rendering	8
44	6 OCCI Text Plain rendering	9
45	6.1 Example	9
46	7 OCCI Header Rendering	9
47	7.1 Example	10
48	8 URI Listing Rendering	10
49	9 Security Considerations	10
50	10 Glossary	11

51	11 Contributors	11
52	12 Intellectual Property Statement	12
53	13 Disclaimer	12
54	14 Full Copyright Notice	12
55	A Change Log	14

1 Introduction

The Open Cloud Computing Interface (OCCI) is a RESTful Protocol and API for all kinds of management tasks. OCCI was originally initiated to create a remote management API for IaaS¹ model-based services, allowing for the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. It has since evolved into a flexible API with a strong focus on interoperability while still offering a high degree of extensibility. The current release of the Open Cloud Computing Interface is suitable to serve many other models in addition to IaaS, including PaaS and SaaS.

In order to be modular and extensible the current OCCI specification is released as a suite of complementary documents, which together form the complete specification. The documents are divided into four categories consisting of the OCCI Core, the OCCI Protocols, the OCCI Renderings and the OCCI Extensions.

- The OCCI Core specification consists of a single document defining the OCCI Core Model. OCCI interaction occurs through *renderings* (including associated behaviors) and is expandable through *extensions*.
- The OCCI Protocol specifications consist of multiple documents, each describing how the model can be interacted with over a particular protocol (e.g. HTTP, AMQP, etc.). Multiple protocols can interact with the same instance of the OCCI Core Model.
- The OCCI Rendering specifications consist of multiple documents, each describing a particular rendering of the OCCI Core Model. Multiple renderings can interact with the same instance of the OCCI Core Model and will automatically support any additions to the model which follow the extension rules defined in OCCI Core.
- The OCCI Extension specifications consist of multiple documents, each describing a particular extension of the OCCI Core Model. The extension documents describe additions to the OCCI Core Model defined within the OCCI specification suite.

The current specification consists of seven documents. This specification describes version 1.2 of OCCI and is backward compatible with 1.1. Future releases of OCCI may include additional protocol, rendering and extension specifications. The specifications to be implemented (MUST, SHOULD, MAY) are detailed in the table below.

Table 1. What OCCI specifications must be implemented for the specific version.

Document	OCCI 1.1	OCCI 1.2
Core Model	MUST	MUST
Infrastructure Model	SHOULD	SHOULD
Platform Model	MAY	MAY
SLA Model	MAY	MAY
HTTP Protocol	MUST	MUST
Text Rendering	MUST	MUST
JSON Rendering	MAY	MUST

2 Notational Conventions

All these parts and the information within are mandatory for implementors (unless otherwise specified). The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

¹Infrastructure as a Service

88 3 Text rendering

89 This document presents the text-based renderings. To be complaint, OCCI implementations MUST implement
90 the three renderings defined in sections 6, 7 and 8.

91 *The following specification of the text-based renderings is in the process of being deprecated and will be
92 removed or significantly changed in the next MAJOR release of the standard.*

93 The document is structured by defining base ABNFs, which can then be combined into renderings, which will
94 be rendered over a protocol (e.g., HTTP) by the specific rendering definitions.

95 4 ABNF Definitions

96 For the following section of 5 these ABNF notations will be used. Implementations MUST hence implement
97 the renderings according to these definitions.

98 4.1 Category ABNF

99 The following syntax MUST be used for **Category** renderings:

```

100 Category           = "Category" ":" #category-value
101   category-value   = term
102                   ";" "scheme" "=" <"> scheme <">
103                   ";" "class" "=" ( class | <"> class <"> )
104                   [ ";" "title" "=" quoted-string ]
105                   [ ";" "rel" "=" <"> type-identifier <"> ]
106                   [ ";" "location" "=" <"> URI <"> ]
107                   [ ";" "attributes" "=" <"> attribute-list <"> ]
108                   [ ";" "actions" "=" <"> action-list <"> ]
109   term              = ( ALPHA | DIGIT ) *( ALPHA | DIGIT | "-" | "_" | "." )
110   scheme            = URI
111   type-identifier   = scheme term
112   class             = "action" | "mixin" | "kind"
113   attribute-list    = attribute-def
114                   | attribute-def *( 1*SP attribute-def )
115   attribute-def     = attribute-name
116                   | attribute-name
117                   | "{" attribute-property *( 1*SP attribute-property ) "}"
118   attribute-property = "immutable" | "required"
119   attribute-name    = term
120   action-list       = action
121                   | action *( 1*SP action )
122   action            = type-identifier

```

123 4.2 Link ABNF

124 The following syntax MUST be used to represent OCCI **Link** type instance references:

```

125 Link               = "Link" ":" #link-value
126   link-value       = "<" URI-reference ">"
127                   ";" "rel" "=" <"> resource-type <">
128                   [ ";" "self" "=" <"> link-instance <"> ]
129                   [ ";" "category" "=" link-type
130                   *( ";" link-attribute ) ]

```

```

131 term           = ( ALPHA | DIGIT ) *( ALPHA | DIGIT | "-" | "_" | "." )
132 scheme        = URI
133 type-identifier = scheme term
134 resource-type  = type-identifier *( 1*SP type-identifier )
135 link-type      = type-identifier *( 1*SP type-identifier )
136 link-instance  = URI-reference
137 link-attribute = attribute-name "=" ( token | quoted-string )
138 attribute-name = term

```

139 The following syntax MUST be used to represent OCCI **Action** instance references:

```

140 ActionLink      = "Link" ":" #link-value
141 link-value      = "<" action-uri ">"
142                 ";" "rel" "=" <"> action-type <">
143 term           = ( ALPHA | DIGIT ) *( ALPHA | DIGIT | "-" | "_" | "." )
144 scheme        = URI
145 type-identifier = scheme term
146 action-type    = type-identifier
147 action-uri     = URI "?" "action=" term

```

148 4.3 Attribute ABNF

```

149 Attribute       = "X-OCCI-Attribute" ":" #attribute-repr
150 attribute-repr  = attribute-name "=" attribute-value
151 attribute-name  = ( ALPHA | DIGIT ) *( ALPHA | DIGIT | "-" | "_" | "." )
152 attribute-value = ( string | number | bool | enum-val )
153 string         = quoted-string
154 number         = (int | float)
155 int            = *DIGIT
156 float         = *DIGIT "." *DIGIT
157 bool          = ("true" | "false")
158 enum-val       = string

```

159 4.4 Location ABNF

```

160 Location        = "X-OCCI-Location" ":" location-value
161 location-value  = URI-reference

```

162 5 Renderings

163 The renderings defined in this section will be used in the specific text rendering defined in section 6 and 7

164 5.1 Entity Instance Rendering

165 Entity instances MUST be rendered according to the following definitions.

166 5.1.1 Resource Instance Rendering

167 A **Resource** instance MUST be rendered using the following definition:

```

168 resource_rendering = 1*( Category CRLF )
169                   *( Link CRLF )
170                   *( Attribute CRLF )

```

171 The rendering of a **Resource** instance MUST represent any associated Action instances using the **ActionLink**
172 **CRLF**.

173 **5.1.1.1 Action Invocation Rendering** Upon an **Action** invocation the client MUST send along the
174 following definition:

```
175   action_definition = 1( Category CRLF )  
176                       *( Attribute CRLF )
```

177 **5.1.2 Link Instance Rendering**

178 A **Link** instance MUST be rendered using the following definition:

```
179   link_rendering = 1*( Category CRLF )  
180                   *( ActionLink CRLF )  
181                   *( Attribute CRLF )
```

182 **5.2 Category Instance Rendering**

183 A **Category** instances MUST be rendered as defined below.

184 **5.2.1 Kind Instance Rendering**

185 A **Kind** instance MUST be rendered as a **Category CRLF**.

186 **5.2.2 Mixin Instance Rendering**

187 A **Mixin** instance MUST be rendered as a **Category CRLF**.

188 **5.2.3 Action Instance Rendering**

189 An **Action** instance MUST be rendered as a **Category CRLF**.

190 Note that an **Action** instance MUST NOT have **Link** and **Actions** references.

191 **5.3 Entity Collection Rendering**

192 A collection of **Resource** or **Link** instances MUST be rendered as following:

```
193   entity_collection_rendering = *( Location CRLF )
```

194 **5.3.1 Resource Collection Rendering**

195 see above

196 **5.3.2 Link Collection Rendering**

197 see above

198 5.4 Category Collection Rendering

199 For the Query interface the following `Category` instance rendering MUST be used:

```
200 category_collection_rendering = *( Category CRLF )
```

201 5.4.1 Kind Collection Rendering

202 see above

203 5.4.2 Mixin Collection Rendering

204 see above

205 5.4.3 Action Collection Rendering

206 see above

207 5.5 Attributes Rendering

208 5.5.1 Entity Instance Attribute Rendering Specifics

209 For Entity instances the following model attribute name to attribute name rendering mappings MUST be used:

Table 2. Entity attribute naming convention

Attribute	Attribute name once rendered
Entity.id	occi.core.id
Entity.title	occi.core.title
Resource.summary	occi.core.summary
Link.target	occi.core.target
Link.target.kind	occi.core.target.kind
Link.source	occi.core.source
Link.source.kind	occi.core.source.kind

210 5.5.2 Mixin Instance Attribute Rendering Specifics

211 When rendering `Mixin.depends` and `Mixin.applies` to the `rel` attribute in the `Category` instance rendering,
 212 only `Mixin.depends` value MUST be used. If `Mixin.depends` contains multiple values, only the first value
 213 MUST be used.

214 5.5.3 Attribute Description Rendering

215 `Attributes` MUST be rendered as defined by the `Attribute` CRLF. If used, the pattern model attribute
 216 MUST be represented as a string in the ERE [2] format.

217 6 OCCI Text Plain rendering

218 The OCCI Text plain rendering specifies a rendering of OCCI instance types in a simple text format.

219 The rendering can be used to render OCCI instances independently of the protocol being used. Thus messages
220 can be delivered by, e.g., the HTTP protocol as specified in [3].

221 The following media-types MUST be used for the OCCI Text plain rendering:

222 `text/occi+plain`

223 and

224 `text/plain`

225 Each entry in the body consists of a name followed by a colon (":") and the field value.

226 6.1 Example

227 The following example show an **Entity** instance rendering using the Text plain rendering.

```
228 < Category: compute; \  
229 <   scheme="http://schemas.ogf.org/occi/infrastructure#" \  
230 <   class="kind";  
231 < Link: </users/foo/compute/b9ff813e-fee5-4a9d-b839-673f39746096?action=start>; \  
232 <   rel="http://schemas.ogf.org/occi/infrastructure/compute/action#start"  
233 < X-OCCI-Attribute: occi.core.id="urn:uuid:b9ff813e-fee5-4a9d-b839-673f39746096"  
234 < X-OCCI-Attribute: occi.core.title="My Dummy VM"  
235 < X-OCCI-Attribute: occi.compute.architecture="x86"  
236 < X-OCCI-Attribute: occi.compute.state="inactive"  
237 < X-OCCI-Attribute: occi.compute.speed=1.33  
238 < X-OCCI-Attribute: occi.compute.memory=2.0  
239 < X-OCCI-Attribute: occi.compute.cores=2  
240 < X-OCCI-Attribute: occi.compute.hostname="dummy"
```

241 7 OCCI Header Rendering

242 The following media-type MUST be used for the OCCI header Rendering:

243 `text/occi`

244 While using this rendering the HTTP Protocol [3] MUST be used and the renderings MUST be placed in the
245 HTTP Header. The body MUST contain the string "OK" on successful operations.

246 The HTTP header fields MUST follow the specification in RFC 7230 [4]. A header field consists of a name
247 followed by a colon (":") and the field value.

248 **Limitations:** HTTP header fields MAY appear multiple times in a HTTP request or response. In order to
249 be OCCI compliant, the specification of multiple message-header fields according to RFC 7230 MUST be
250 fully supported. In essence there are two valid representations of multiple HTTP header field values. A header
251 field might either appear several times or as a single header field with a comma-separated list of field values.
252 Due to implementation issues in many web frameworks and client libraries it is RECOMMENDED to use the
253 comma-separated list format for best interoperability.

254 HTTP header field values, which contain separator characters, MUST be properly quoted according to RFC 7230.

255 Space in the HTTP header section of a HTTP request is a limited resource. By this, it is noted that many
256 HTTP servers limit the number of bytes that can be placed in the HTTP header area. Implementers MUST
257 be aware of this limitation in their own implementations and take appropriate measures so that truncation of
258 header data does NOT occur.

259 7.1 Example

260 The following example shows an **Entity** instance rendering using the Text header rendering.

```
261 < Category: compute; \  
262     scheme="http://schemas.ogf.org/occi/infrastructure#" \  
263     class="kind";  
264 < Link: </users/foo/compute/b9ff813e-fee5-4a9d-b839-673f39746096?action=start>; \  
265     rel="http://schemas.ogf.org/occi/infrastructure/compute/action#start"  
266 < X-OCCE-Attribute: occi.core.id="urn:uuid:b9ff813e-fee5-4a9d-b839-673f39746096", \  
267     occi.core.title="My Dummy VM", occi.compute.architecture="x86", \  
268     occi.compute.state="inactive", occi.compute.speed=1.33, \  
269     occi.compute.memory=2.0, occi.compute.cores=2, \  
270     occi.compute.hostname="dummy"  
271 < OK
```

272 8 URI Listing Rendering

273 The following media-types MUST be used for the URI Rendering:

274 `text/uri-list`

275 This rendering cannot render resource instances or Kinds or Mixins directly but just links to them. For concrete
276 rendering of Kinds and Categories the Content-types `text/occi`, `text/plain` MUST be used. If a request is
277 done with the `text/uri-list` in the Accept header, while not requesting for a Listing a Bad Request MUST
278 be returned. Otherwise a list of resources MUST be rendered in `text/uri-list` format as defined in [5],
279 which can be used for listing resource in collections or the name-space of the OCCE implementation.

280 9 Security Considerations

281 OCCE does not require that an authentication mechanism be used nor does it require that client to service
282 communications are secured. It does RECOMMEND that an authentication mechanism be used and that
283 where appropriate, communications are encrypted using HTTP over TLS. The authentication mechanisms
284 that MAY be used with OCCE are those that can be used with HTTP and TLS. For further discussion see the
285 appropriate section in [3].

286 10 Glossary

Term	Description
Action	An OCCI base type. Represents an invocable operation on an Entity sub-type instance or collection thereof.
Attribute	A type in the OCCI Core Model. Describes the name and properties of attributes found in Entity types.
Category	A type in the OCCI Core Model and the basis of the OCCI type identification mechanism. The parent type of Kind .
capabilities	In the context of Entity sub-types capabilities refer to the Attributes and Actions exposed by an entity instance .
Collection	A set of Entity sub-type instances all associated to a particular Kind or Mixin instance.
Entity	An OCCI base type. The parent type of Resource and Link .
entity instance	An instance of a sub-type of Entity but not an instance of the Entity type itself. The OCCI model defines two sub-types of Entity : the Resource type and the Link type. However, the term <i>entity instance</i> is defined to include any instance of a sub-type of Resource or Link as well.
Kind	A type in the OCCI Core Model. A core component of the OCCI classification system.
287 Link	An OCCI base type. A Link instance associates one Resource instance with another.
Mixin	A type in the OCCI Core Model. A core component of the OCCI classification system.
mix-in	An instance of the Mixin type associated with an <i>entity instance</i> . The “mix-in” concept as used by OCCI <i>only</i> applies to instances, never to Entity types.
OCCI	Open Cloud Computing Interface.
OGF	Open Grid Forum.
Resource	An OCCI base type. The parent type for all domain-specific Resource sub-types.
resource instance	See <i>entity instance</i> . This term is considered obsolete.
tag	A Mixin instance with no attributes or actions defined. Used for taxonomic organisation of entity instances.
template	A Mixin instance which if associated at instance creation-time pre-populate certain attributes.
type	One of the types defined by the OCCI Core Model. The Core Model types are Category , Attribute , Kind , Mixin , Action , Entity , Resource and Link .
concrete type/sub-type	A concrete type/sub-type is a type that can be instantiated.
URI	Uniform Resource Identifier.
URL	Uniform Resource Locator.
288 URN	Uniform Resource Name.

289 11 Contributors

290 We would like to thank the following people who contributed to this document:

Name	Affiliation	Contact
Michael Behrens	R2AD	behrens.cloud at r2ad.com
Mark Carlson	Toshiba	mark at carlson.net
Augusto Ciuffoletti	University of Pisa	augusto.ciuffoletti at gmail.com
Andy Edmonds	ICCLab, ZHAW	edmo at zhaw.ch
Sam Johnston	Google	samj at samj.net
Gary Mazzaferro	Independent	garymazzaferro at gmail.com
Thijs Metsch	Intel	thijs.metsch at intel.com
291 Ralf Nyrén	Independent	ralf at nyren.net
Alexander Papaspyrou	Adesso	alexander at papaspyrou.name
Boris Parák	CESNET	parak at cesnet.cz
Alexis Richardson	Weaveworks	alexis.richardson at gmail.com
Shlomo Swidler	Orchestratus	shlomo.swidler at orchestratus.com
Florian Feldhaus	Independent	florian.feldhaus at gmail.com
Zdeněk Šustr	CESNET	zdenek.sustr at cesnet.cz
Jean Parpaillon	Inria	jean.parpaillon at inria.fr
Philippe Merle	Inria	philippe.merle@inria.fr

292 Next to these individual contributions we value the contributions from the OCCI working group.

293 12 Intellectual Property Statement

294 The OGF takes no position regarding the validity or scope of any intellectual property or other rights that
 295 might be claimed to pertain to the implementation or use of the technology described in this document or the
 296 extent to which any license under such rights might or might not be available; neither does it represent that
 297 it has made any effort to identify any such rights. Copies of claims of rights made available for publication
 298 and any assurances of licenses to be made available, or the result of an attempt made to obtain a general
 299 license or permission for the use of such proprietary rights by implementers or users of this specification can be
 300 obtained from the OGF Secretariat.

301 The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications,
 302 or other proprietary rights which may cover technology that may be required to practice this recommendation.
 303 Please address the information to the OGF Executive Director.

304 13 Disclaimer

305 This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all
 306 warranties, express or implied, including but not limited to any warranty that the use of the information herein
 307 will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

308 14 Full Copyright Notice

309 Copyright © Open Grid Forum (2009-2016). All Rights Reserved.

310 This document and translations of it may be copied and furnished to others, and derivative works that comment
 311 on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in
 312 whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph
 313 are included on all such copies and derivative works. However, this document itself may not be modified in
 314 any way, such as by removing the copyright notice or references to the OGF or other organizations, except
 315 as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights
 316 defined in the OGF Document process must be followed, or as required to translate it into languages other
 317 than English.

318 The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or
 319 assignees.

320 **References**

- 321 [1] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119 (Best Current Practice),
322 Internet Engineering Task Force, Mar. 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2119.txt>
- 323 [2] "Extended Regular Expressions," The Open Group, 1997. [Online]. Available: [http://pubs.opengroup.org/
324 onlinepubs/7908799/xbd/re.html](http://pubs.opengroup.org/onlinepubs/7908799/xbd/re.html)
- 325 [3] R. Nyren, T. Metsch, and A. Edmonds, "Open Cloud Computing Interface – HTTP Protocol," Draft,
326 November 2015. [Online]. Available: TBD
- 327 [4] R. Fielding and J. Gettys, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," RFC
328 7230, Internet Engineering Task Force, Jun. 2014. [Online]. Available: <http://www.ietf.org/rfc/rfc7230.txt>
- 329 [5] M. Mealling and J. R. Daniel, "URI Resolution Services Necessary for URN Resolution," RFC 2483,
330 Internet Engineering Task Force, Jan. 1999. [Online]. Available: <https://tools.ietf.org/html/rfc2483>

331 **A Change Log**

332 The corrections introduced by the February 23, 2016 update are summarized below. This section describes the
333 possible impact of the corrections on existing implementations and associated dependent specifications.

- 334 • Relaxed rules on `term` values allowing the use of: alphanumerical characters (a-zA-Z0-9), “_”, “-” and
335 “.”.
- 336 • Explicitly stated how `Mixin.depends` and `Mixin.applies` should be rendered to `rel` on `Mixin` instances.