GWD-R Ralf Nyrén

2 OCCI-WG

February 4, 2013

Updated: January 21, 2015

Open Cloud Computing Interface - RESTful HTTP Protocol

- 6 Status of this Document
- This is a draft proposal of a new document in the Open Cloud Computing Interface specification suite.
- 8 Copyright Notice
- 9 Copyright ©Open Grid Forum (2013-2014). All Rights Reserved.
- 10 Trademarks
- 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
- requirements and focuses on the scope of important capabilities required to support modern service offerings.
- 16 Comments

17 • ...

18 Contents

19	1	Introduction				
20	2	Notational Conventions				
21	3	OCCI RESTful HTTP Protocol				
22	4	Namespace				
23		4.1 Bound and unbound paths				
24	5	Headers and status code	Headers and status codes			
25		5.1 Mandatory HTTP re	Mandatory HTTP requests headers			
26		5.2 Mandatory HTTP re	Mandatory HTTP response headers			
27		5.3 HTTP status codes	HTTP status codes			
28	6	HTTP methods applied	HTTP methods applied to entity instance URLs			
29		6.1 GET entity instance		. 5		
30		6.1.1 Client GET re	equest	. 5		
31		6.1.2 Server GET r	response	. 6		
32		6.2 PUT entity instance		. 6		
33		6.2.1 Client PUT r	equest	. 6		
34		6.2.2 Server PUT i	response	. 6		
35		6.3 POST entity instance	e (action)	. 7		
36		6.3.1 Client POST	action request	. 7		
37		6.3.2 Server POST	action response	. 7		
38			e			
39		6.5 DELETE entity insta	nnce	. 8		
40		-	TE request			
41			TE response			
42	7	HTTP methods applied to collections URLs				
43		7.1 GET collection		. 8		
44		7.1.1 Client GET re	equest	. 9		
45		7.1.2 Server GET r	response	. 9		
46		7.2 POST collection		. 10		
47		7.2.1 Client POST	request	. 10		
48		7.2.2 Server POST	response	. 10		
49		7.3 POST collection with	h "action" query parameter	. 11		
50						
51						
52	8	HTTP methods applied to QI 11				
53	9	Glossary 1				
		-				

	GWD-R	January 21, 2015
54	10 Intellectual Property Statement	12
55	11 Disclaimer	13
56	12 Full Copyright Notice	13

7 1 Introduction

in RFC 2119 [?].

62

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

The following terms [?] are used when referring to URI components:

3 OCCI RESTful HTTP Protocol

- This document specifies the OCCI HTTP Protocal, a RESTful protocol for communication between OCCI
- ₇₀ Server and OCCI Client. The OCCI HTTP Protocol support multiple different data formats as payload. Data
- 71 formats are specified an separate documents.
- 72 TBD: general intro to REST etc

73 4 Namespace

- 74 The OCCI HTTP Protocol maps the OCCI Core model into the URL hierarchy by binding Kind and Mixin
- rs instances to unique URL paths. Such a URL path is called the location of the Kind or Mixin. A provider is
- ₇₆ free to choose the *location* as long as it is unique within the service provider's URL namespace. For example,
- the Kind instance for the Compute type may be bound to /my/occi/api/compute/.
- A Kind instance whose associated type cannot be instantiated MUST NOT be bound to an URL path. This
- ₇₉ applies to the Kind instance for OCCI Entity which, according to OCCI Core, cannot be instantiated [?].
- 80 TODO: sub resources here.

4.1 Bound and unbound paths

- 82 Since a limited set of URL paths are bound to Kind and Mixin instances the URL hierarchy consists of both
- bound and unbound paths. A bound URL path is the location of a Kind or Mixin collection.
- 84 An unbound URL path MAY represent the union of all Kind and Mixin collection "below" the unbound path.
- 85 TODO: FIXME: Should this be a MUST instead?

5 Headers and status codes

- 87 TODO: add all other HTTP niftyness from old spec here.
- 88 OCCI Clients and Servers must include a minimum set of mandatory HTTP headers in each request and
- 89 response in order to be compliant. There is also a minimum set of HTTP status codes which must be
- 90 supported by an implementation of the OCCI HTTP Protocol.

¹http://schemas.ogf.org/occi/infrastructure#compute

5.1 Mandatory HTTP requests headers

Accept An OCCI Client SHOULD specify the media-types the OCCI data formats it supports in the Accept header.

Content-type If an OCCI Client submits payload in a HTTP request the OCCI Client MUST specify the media-type of the OCCI data format in the Content-type header.

5.2 Mandatory HTTP response headers

- Content-type An OCCI Server MUST specify the media-type of the OCCI data format used in a HTTP Response.
- Server An OCCI Server MUST specify the OCCI HTTP Protocol version number.

5.3 HTTP status codes

- The below list specifies the minimum set of HTTP status codes an OCCI Client MUST understand. An OCCI Server MAY return other HTTP status codes but the exact client behaviour in such cases is not specified.
- 103 **200**
- 104 **204**
- 105 301
- 106 400
- 107 403
- 108 404
- 109 405

₁₁₀ 6 HTTP methods applied to entity instance URLs

- 111 TODO: Sanity check this section against v1.1...
- This section describes the HTTP methods used to retrieve and manipulate individual entity instances. An entity instance refers to an instance of the OCCI Resource type, OCCI Link type or a sub-type thereof [?].
- Each HTTP method described is assumed to operate on an URL referring to a single element in a collection, an URL such as the following:
- http://example.com/compute/012d2b48-c334-47f2-9368-557e75249042

117 6.1 GET entity instance

118 The HTTP GET method retrieves a representation of a single (existing) entity instance.

119 6.1.1 Client GET request

- 120 The body of the HTTP GET request MUST be empty.
- 121 GET /compute/012d2b48-c334-47f2-9368-557e75249042 HTTP/1.1
- 122 Host: example.com
- 123 Accept: application/occi+xxx
- User-Agent: occi-client/x.x OCCI/1.1

5 6.1.2 Server GET response

126 The body of the HTTP GET response MUST contain a representation of the entity instance.

```
HTTP/1.1 200 OK
Server: occi-server/x.x OCCI/1.1
Content-Type: application/occi+xxx; charset=utf-8
```

132 6.2 PUT entity instance

The HTTP PUT method either *creates* a new or *replaces* an existing entity instance at the specified URL.

The unique identifier of the entity instance (Entity.id) MUST be specified in the request URL. An OCCI
Client MAY also specify Entity.id in the payload however it MUST be identical to the identifier specified as part of the request URL.

137 If an entity instance with the specified identifier does not exist the PUT request allows the OCCI Client to choose the Entity.id of the new instance. An OCCI Server MAY refuse such a request with HTTP status code 405. This would indicate that the OCCI Server does not allow user defined entity identifiers.

The PUT method MUST be idempotent, i.e. multiple identical PUT requests should have the same effect as a single request.

142 6.2.1 Client PUT request

The full representation of the entity instance MUST be supplied in the HTTP body of the request. The request body MUST only include a representation of a single entity instance.

If the request represent an OCCI Resource (as opposed to an OCCI Link) the representation MUST NOT
 include any Link instances associated with the Resource instance. A server MUST refuse a request including
 associated Links

Any OCCI Links associated with an existing OCCI Resource MUST be left intact.

```
PUT /compute/012d2b48-c334-47f2-9368-557e75249042 HTTP/1.1
149
   Host: example.com
   Accept: application/occi-entity+json
151
   User-Agent: occi-client/x.x OCCI/1.1
152
   Content-Type: application/occi-entity+json; charset=utf-8
153
155
      "kind": "http://schemas.ogf.org/occi/infrastructure#compute",
156
      "mixins": [ ... ],
157
      "attributes": { ... }
   }
159
```

6.2.2 Server PUT response

161 Content-Type application/occi-entity+json

Upon success an OCCI server MUST return HTTP status code 200 and a complete JSON representation of the created/replaced entity instance in single-entity-instance format. The response MUST be identical² to that of a subsequent GET request of the same URL.

²Provided the entity instance was not changed in the meantime.

6.3 POST entity instance (action)

177 There are two methods to invoke an OCCI Action using the JSON Rendering.

- 1. Supply the query parameter "action" together with the request. The value of "action" MUST be the term of the action Category.
- 2. Specify application/occi-action+json in the Content-Type header and supply a request payload formatted according to section ??. In order to specify action attributes this method MUST be used.

An OCCI Client MAY combine the two methods if the "action" parameter's value is equal to the Category term in the body.

84 6.3.1 Client POST action request

```
Accept application/occi-entity+json
```

Content-Type application/occi-action+json

187 The example shows the combined method.

```
POST /compute/012d2b48-c334-47f2-9368-557e75249042?action=stop HTTP/1.1
   Host: example.com
189
   Accept: application/occi-entity+json
   User-Agent: occi-client/x.x OCCI/1.1
   Content-Type: application/occi-action+json; charset=utf-8
192
193
   {
194
      "category": "http://schemas.ogf.org/occi/infrastructure/compute/action#stop",
195
      "attributes": {
196
        "method": "graceful"
197
     }
   }
199
```

6.3.2 Server POST action response

201 Content-Type application/occi-entity+json

If the request Accept header contains application/occi-entity+json the server MAY return status code 200 and a full representation of the entity instance. Otherwise the server MUST return status code 204 and no response payload.

```
205 HTTP/1.1 204 OK
206 Server: occi-server/x.x OCCI/1.1
```

of 6.4 POST entity instance

208 Content-Type application/occi-entity+json

TODO: This would imply a partial update of the entity instance. While it is easy to supply only the attributes to be updated the question is if there are any valid use cases for partial updates using JSON?

211 6.5 DELETE entity instance

The HTTP DELETE method destroys an entity instance and any OCCI Links associated with an OCCI Resource.

214 6.5.1 Client DELETE request

```
215 Content-Type application/occi-entity+json
```

```
DELETE /compute/012d2b48-c334-47f2-9368-557e75249042 HTTP/1.1 Host: example.com
Accept: application/occi-entity+json
```

219 6.5.2 Server DELETE response

```
220 HTTP/1.1 204 OK
221 Server: occi-server/x.x OCCI/1.1
```

7 HTTP methods applied to collections URLs

```
TODO: NOT fully updated yet!
```

This section describes the HTTP methods used to manipulate collections. Each HTTP method described is assumed to operate on an URL referring to a collection of elements, an URL such as the following:

```
http://example.com/storage/
```

A collection consist of a set of entity instances and there are three different types of collections which may be exposed by an OCCI server. The request and response format is identical for all three types collections although the semantics differ slightly for the PUT and POST methods.

Kind locations The location associated with an OCCI Kind instance represents the collection of all entity instances of that particular Kind.

Mixin locations The location of an OCCI Mixin instance represents the collection of all entity instances associated with that Mixin.

Arbitrary path Any path in the URL namespace which is neither a Kind nor a Mixin location. A typical example is the root URL e.g. http://example.com/. Such a path combines all collections in the sub-tree starting at the path. Therefore the root URL is a collection of all entity instances available.

7.1 **GET** collection

The HTTP GET method retrieves a list of all entity instances in the collection. Filtering and pagination information is encoded in the query string of the URL.

7.1.1 Client GET request

The query string of the request URL MUST have the following format:

```
query-string
242
                         | "?" query-parameter *( "&" query-parameter )
243
     query-parameter
                         = attribute-filter
                         | category-filter
245
                         | pagination-marker
246
                         | pagination-limit
247
                        = "q=" attribute-search *( "+" attribute-search )
     attribute-filter
                        = 1*( string-urlencoded |
     attribute-search
249
                               attribute-name "%3D" string-urlencoded )
250
     category-filter
                        = "category=" string-urlencoded
251
     pagination-marker = "marker=" UUID
252
     pagination-limit = "limit=" 1*( DIGIT )
253
     attribute-name
                        = attr-component *( "." attr-component )
254
     attr-component
                         = LOALPHA *( LOALPHA | DIGIT | "-" | "_" )
     string-urlencoded = *( ALPHA | DIGIT | "-" | "_" | "." | "~" | "%" )
256
257
```

258 TODO: FIXME: UUID in ABNF

Filtering A search filter can be applied to categories and attributes of entity instances in a collection. An OCCI server SHOULD support filtering. The query parameters MUST be URL encoded.

Attribute filters are specified using the *q* query parameter. A filter such as q=ubuntu+inactive would match all entity instances whose combined set of attribute values includes both the word "ubuntu" and "inactive". It is also possible to match on specific attributes by preceding the search term with the attribute name and an equal sign, for example occi.core.title%3Dubuntu+occi.compute.state%3Dinactive.

The category filter is specified using the *category* query parameter and represent a single Kind, Mixin or
Action category to be matched. The following query would include only entity instances of the Compute type:
category=http%3A%2F%2Fschemas.ogf.org%2Focci%2Finfrastructure%23compute

Pagination TODO: FIXME: marker instead of start An OCCI client MAY request that the server only return a subset of a collection. This is accomplished using the *marker* and *limit* query parameters. An OCCI server MUST support pagination.

The marker parameter specifies the offset into the collection. A value of zero, marker=0 indicates the beginning of the collection. The limit parameter sets the maximum number of elements to include in the response. For example ?marker=...&limit=10 would indicate the third page with a limit of 10 elements per page.

275 Example request

280

```
276  GET /storage/?q=ubuntu+server&limit=20 HTTP/1.1
277  Host: example.com
278  Accept: application/occi-collection+json
279  User-Agent: occi-client/x.x OCCI/1.1
```

7.1.2 Server GET response

```
HTTP/1.1 200 OK
Server: occi-server/x.x OCCI/1.1
Content-Type: application/occi-collection+json; charset=utf-8
```

```
284
285
      "collection": [
286
        {
287
           "kind": "..." ,
288
           "mixins": [ ... ],
289
           "actions": [ ... ],
290
           "links": [ ... ],
291
           "attributes": { ... },
        },
293
        { ... },
294
        { ... }
295
      ],
296
      "limit": 20,
297
      "size": 137,
298
      "next": "http://example.com/storage/?q=ubuntu+server&marker=59b50...9b3&limit=20"
299
   }
```

7.2 POST collection

The HTTP POST method is used to create/update one or more entity instances in a single atomic request.

An OCCI server MUST identify existing entity instances using the occi.core.id attribute.

7.2.1 Client POST request

```
POST /storage/ HTTP/1.1
305
   Host: example.com
306
   Accept: application/occi-collection+json
307
   User-Agent: occi-client/x.x OCCI/1.1
   Content-Type: application/occi-collection+json; charset=utf-8
309
310
   {
311
      "collection": [
312
        {
313
          "kind": "...",
314
          "mixins": [ ... ],
315
          "links": [ ... ],
316
          "attributes": { ... },
317
        },
318
        { ... },
319
        { ... }
      ]
321
   }
322
```

7.2.2 Server POST response

323

```
HTTP/1.1 204 OK
Server: occi-server/x.x OCCI/1.1
```

TODO: Should we support HTTP 200 returning the whole collection? Or maybe just the entity instances created/updated?

28 7.3 POST collection with "action" query parameter

329 todo

7.4 PUT collection

Replace the entire collection with a new one. TODO: Should we support this?

332 7.5 DELETE collection

Delete the entire collection. TODO: Should we support this?

8 HTTP methods applied to QI

TODO: write this.

9 Glossary

	Term	Description
	Action	An OCCI base type. Represents an invocable operation on a Entity sub-type in-
		stance 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
	Category	mechanism. The parent type of Kind.
	capabilities	In the context of Entity sub-types capabilities refer to the OCCI Attributes and
	capabilities	OCCI Actions exposed by an entity instance .
	Client	An OCCI client.
	Collection	A set of Entity sub-type instances all associated to a particular Kind or Mixin
	F	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.
	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"
337		concept as used by OCCI <i>only</i> applies to instances, never to Entity types.
	model attribute	An internal attribute of a the Core Model which is <i>not</i> client discoverable.
	OCCI	Open Cloud Computing Interface.
	OCCI base type	One of Entity, Resource, Link or Action.
	OCCI Action	see Action.
	OCCI Attribute	A client discoverable attribute identified by an instance of the Attribute type.
	Occi Attribute	Examples are occi.core.title and occi.core.summary.
	OCCI Catagony	see Category.
	OCCL Entity	
	OCCI Entity OCCI Kind	see Entity. see Kind.
	OCCI Link	see Link.
	OCCI Mixin	see Mixin.
	OGF	Open Grid Forum.
	Resource	An OCCI base type. The parent type for all domain-specific Resource sub-types.
	resource instance	See entity instance. This term is considered obsolete.
	tag	A Mixin instance with no attributes or actions defined.
	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.
	URN	Uniform Resource Name.
338		1

10 Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the

extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

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

350 11 Disclaimer

356

357

358

360

361

362

363

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

12 Full Copyright Notice

Copyright © Open Grid Forum (2009-2014). All Rights Reserved.

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

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