Ralf Nyrén

September 16, 2011 Updated: February 21, 2012

Open Cloud Computing Interface - JSON Rendering

Status of this Document

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

Copyright Notice

Copyright ©Open Grid Forum (2012). All Rights Reserved.

Trademarks

OCCI is a trademark of the Open Grid Forum.

Abstract

This document, part of a document series, produced by the OCCI working group within the Open Grid Forum (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.

<u>Comments</u>

Contents	
----------	--

1	Intro	oduction	3
2	Nota	ational Conventions	3
3	JSO	IN representation	4
	3.1	Mandatory HTTP headers	4
	3.2	Introduction	4
	3.3	Syntax and Semantics of the JSON Rendering	4
		3.3.1 Rendering of the OCCI Category, Kind and Mixin Types	4
		3.3.2 Rendering of OCCI Entity Attributes	6
4	ΗΤΊ	TP methods applied to resource instance URLs	8
	4.1	GET resource instance	8
		4.1.1 Client GET request	8
		4.1.2 Server GET response	8
	4.2	PUT resource instance	8
		4.2.1 Client PUT request	8
		4.2.2 Server PUT response	9
	4.3	POST resource instance with "action" query parameter	9
		4.3.1 Client POST action request	9
		4.3.2 Server POST action response	10
	4.4	POST resource instance without any query parameters	10
	4.5	DELETE resource instance	10
		4.5.1 Client DELETE request	10
		4.5.2 Server DELETE response	10
5	нтт	TP methods applied to collections URLs	10
	5.1	GET collection	11
		5.1.1 Client GET request	11
		5.1.2 Server GET response	12
	5.2	POST collection	12
		5.2.1 Client POST request	12
		5.2.2 Server POST response	12
	5.3	POST collection with "action" query parameter	13
	5.4	PUT collection	13
	5.5	DELETE collection	13
6	Mor	re examples	13

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 complimentary documents, which together form the complete specification. The documents are divided into three categories consisting of the OCCI Core, the OCCI Renderings and the OCCI Extensions.

- The OCCI Core specification consists of a single document defining the OCCI Core Model. The OCCI Core Model can be interacted with *renderings* (including associated behaviours) and expanded through *extensions*.
- 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. They do not require changes to the HTTP Rendering specifications as of this version of the specification.

The current specification consists of three documents. This specification describes version 1.1 of OCCI. Future releases of OCCI may include additional rendering and extension specifications. The documents of the current OCCI specification suite are:

OCCI Core describes the formal definition of the the OCCI Core Model [?].

- **OCCI HTTP Rendering** defines how to interact with the OCCI Core Model using the RESTful OCCI API [?]. The document defines how the OCCI Core Model can be communicated and thus serialised using the HTTP protocol.
- **OCCI Infrastructure** contains the definition of the OCCI Infrastructure extension for the IaaS domain [?]. The document defines additional resource types, their attributes and the actions that can be taken on each resource type.

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 [?].

This document uses the Augmented Backus-Naur Form (ABNF) notation of RFC 2616 [?], and explicitly includes the following rules from it: quoted-string, token, SP (space), LOALPHA, DIGIT.

¹Infrastructure as a Service

3 JSON representation

3.1 Mandatory HTTP headers

HTTP headers which MUST be present in every JSON-rendered request and response.

Mandatory HTTP request headers:

Accept: application/occi+json

Content-Type: application/occi+json

Mandatory HTTP response headers:

Content-Type: application/occi+json

3.2 Introduction

The OCCI HTTP Rendering uses many of the features the HTTP and underlying protocols offer and builds upon the Resource Oriented Architecture (ROA). ROA's use Representation State Transfer (REST) [?] to cater for client and service interactions. These interactions are detailed in the OCCI HTTP Rendering document.

3.3 Syntax and Semantics of the JSON Rendering

All data transferred using the *application/occi+json* content type is structured text. These rendering structures are compliant with and follow the rules of HTTP headers [?]. Four specific rendering structures are only ever used:

- Category
- Link
- X-OCCI-Attribute
- X-OCCI-Location

The *application/occi+json* content type renders the rendering structures in the body of the HTTP request/response according to RFC 4627.

3.3.1 Rendering of the OCCI Category, Kind and Mixin Types

The following terms defined in ABNF will be used for the specification of the JSON category rendering:

term	=	LOALPHA *(LOALPHA DIGIT "-" "_")
scheme	=	URI
type-identifier	=	scheme term
attribute-name	=	<pre>attr-component *("." attr-component)</pre>
attr-component	=	LOALPHA *(LOALPHA DIGIT "-" "_")
action	=	type-identifier

All category renderings MUST be rendered as a JSON Object with one or more of the following entries:

Field Name	Туре	Description	Requirement
kinds	JSON Array	List of all OCCI Kinds	Mandatory ²
mixins	JSON Array	List of all OCCI Mixins	Mandatory ³
categories	JSON Array	List of all OCCI Categories (e.g. categories for describing	Mandatory ⁴
		Actions)	

Each entry in the category, kind and mixin lists MUST be rendered as

Field Name	Туре	Description	Requirement
term	JSON String	Value MUST be rendered as term	Mandatory
scheme	JSON String	Value MUST be rendered as scheme	Mandatory
title	JSON String	Value MAY contain any character. Quotes inside the	Optional
		value MUST be escaped.	
related	JSON String	Value MUST be rendered as type-identifier	Optional
location	JSON String	Location MUST be rendered as URI	Optional
attributes	JSON Object	key/value pairs of attributes and their description	Optional
actions	JSON Array	List of action type identifiers	Optional

The attribute key/value pairs are rendered as

Field Name	Туре	Description	Requirement
attribute-name	JSON Object	Key MUST be rendered according to attribute-name . Value MUST be rendered according to <i>Attribute property rendering</i>	Required

Attribute property rendering

Field Name	Туре	Description	Requirement
mutable	true or false	true when attribute is modifiable by the client	Mandatory
required	true or false	true if attribute MUST be specified by the client	Mandatory
type	JSON String	MUST be one of the following types: string integer float boolean	Mandatory
range	JSON String	Specify a range for the attribute value. The range should either be a list of allowed entries enclosed in curly brackets and separated by a pipe (e.g. $\{x86 x64\}$) or alternatively two numbers sparated by two dots. For strings the minimum and maximum length should be specified (e.g. 412 characters), for integers and floats the minimum and maximum values (e.g. 14 or - 2.55.0) should be specified	Optional
default	JSON String	The default value should be specified here. This can be used for specifying a provider default for kinds or to create mixin templates.	Optional

The following example illustrates a rendering of the Kind instance assigned to the Compute type [?]:

```
"kinds": [
    {
        "term": "compute",
        "scheme": "http://schemas.ogf.org/occi/infrastructure#",
        "title": "Compute Resource",
        "related": "http://schemas.ogf.org/occi/core#resource",
        "attributes": {
            "occi.compute.architecture": {
                "mutable": true,
                "required": false,
```

{

 $^{^2}$ if filtering is used, only the filtered categories are mandotry $^3 \text{see}$ above

```
"type": "string",
          "range": "{x86|x64}",
          "default": "x86"
        },
        "occi.compute.cores": {
          "mutable": true,
          "required": false,
          "type": "integer",
          "range": "1..16",
          "default": "1"
        },
        "occi.compute.hostname": {
          "mutable": true,
          "required": false,
          "type": "string"
          "range": "1..256",
        },
        "occi.compute.speed": {
          "mutable": true,
          "required": false,
          "type": "float",
          "range": "1.0..2.8",
          "default": "2.2"
        },
        "occi.compute.memory": {
          "mutable": true,
          "required": false,
          "type": "float"
          "range": "1..16",
          "default": "4"
        },
        "occi.compute.state": {
          "mutable": false,
          "required": false,
          "type": "string"
          "range": "{active|inactive|suspended}",
          "default": "inactive"
        }
      },
      "actions": [
        "http://schemas.ogf.org/occi/infrastructure/compute/action#start",
        "http://schemas.ogf.org/occi/infrastructure/compute/action#stop",
        "http://schemas.ogf.org/occi/infrastructure/compute/action#restart",
        "http://schemas.ogf.org/occi/infrastructure/compute/action#suspend"
      ].
      "location": "/api/compute/"
    }
]
```

3.3.2 Rendering of OCCI Entities

Entities MUST be rendered as JSON Objects containing the following entries:

}

Field Name	Туре	Description	Requirement
kind	JSON Object	Short category rendering	Mandatory
mixins	JSON Array	Array of short category renderings	Optional
actions	JSON Array	List of applicable actions as JSON Objects	Mandatory
attributes	JSON Object	key/value pairs of attributes	Mandatory
links	JSON Array	list of links as JSON Objects	Mandatory
location	JSON String	URI	Mandatory

The short category rendering must be rendered as follows:

Field Name	Туре	Description	Requirement
term	JSON String	term	Mandatory
scheme	JSON String	scheme	Mandatory

Each entry of the list of applicable actions MUST be rendered as:

Field Name	Туре	Description	Requirement
title	JSON String	Title of the action	Optional
uri	JSON String	URI	Mandatory
type	JSON String	type-identifier	Mandatory

Each attribute MUST be rendered as:

Field Name	Туре	Description	Requirement
attribute-name	JSON String or JSON Number	Value of the attribute	Mandatory

Each entry of the list of links MUST be rendered as:

Field Name	Туре	Description	Requirement
title	JSON String	Title of the Link	Optional
target	JSON String	URI	Mandatory
target_type	JSON String	URI	Mandatory
location	JSON String	URI	Mandatory
type	JSON String	URI	Mandatory
attributes	JSON Object	key/value pairs of attributes	Mandatory

```
{
```

}

```
"kind": {
    "term": "compute",
    "scheme": "http://schemas.ogf.org/occi/infrastructure#",
},
"actions": [
    {
        "title": "Start Compute Resource",
        "uri": "/api/compute/3e09b631-dc81-4495-b307-dca15e14c374?action=start",
        "type": "http://schemas.ogf.org/occi/infrastructure/compute/action#start"
    }
],
"attributes": {
    "occi.core.id": "3e09b631-dc81-4495-b307-dca15e14c374",
    "occi.compute.architecture": "x86_64",
    "occi.compute.speed": 2.6699999999999999,
    "occi.compute.memory": 1.0,
    "occi.compute.state": "inactive"
}
```

4 HTTP methods applied to resource instance URLs

This section describes the HTTP methods used to retrieve and manipulate individual resource instances. 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

An OCCI client MUST supply the mandatory headers described in section 3.1 with every HTTP request.

4.1 GET resource instance

The HTTP GET method retrieves the JSON representation of an OCCI resource instance.

4.1.1 Client GET request

The body of the HTTP GET request MUST be empty.

```
GET /compute/012d2b48-c334-47f2-9368-557e75249042 HTTP/1.1
Host: example.com
Accept: application/occi+json
User-Agent: occi-client/x.x OCCI/1.1
```

4.1.2 Server GET response

```
HTTP/1.1 200 OK
Server: occi-server/x.x OCCI/1.1
Category: compute;
        scheme="http://schemas.ogf.org/occi/infrastructure#";
        class="kind";
        title="Compute Resource"
Content-Type: application/occi+json; charset=utf-8
{
        "kind": { ... },
        "mixins": [ ... ],
        "actions": [ ... ],
        "links": [ ... ],
        "links": [ ... ],
        "attributes": { ... },
        "location": "http://example.com/compute/012d2b48-c334-47f2-9368-557e75249042"
}
```

4.2 PUT resource instance

The HTTP PUT method creates or replaces the resource instance at the specified URL. Since the resource identifier is supplied in the request URL an OCCI server MAY refuse to create a new instance.

4.2.1 Client PUT request

The full JSON representation of the resource instance MUST be supplied in the HTTP body of the request. RN: Including Links in the request breaks the rule of PUT being idempotent. Simply prohibit Links in PUT requests?

4.2.2 Server PUT response

Upon success an OCCI server MUST return HTTP status code 200 and a complete JSON representation of the created/replaced resource instance. The response MUST be identical⁵ of the to that of a subsequent GET request same URL.

4.3 POST resource instance with "action" query parameter

An OCCI Action is invoked using the HTTP POST method together with query parameter named "action".

4.3.1 Client POST action request

```
POST /compute/012d2b48-c334-47f2-9368-557e75249042?action=stop HTTP/1.1
Host: example.com
Accept: application/occi+json
User-Agent: occi-client/x.x OCCI/1.1
Content-Type: application/occi+json; charset=utf-8
{
    "action": {
    "term": "stop",
    "scheme": "http://schemas.ogf.org/occi/infrastructure/compute/action#",
    },
    "attributes": {
```

⁵Provided the resource instance was not changed in the meantime.

```
"method": "graceful"
},
}
```

4.3.2 Server POST action response

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

4.4 POST resource instance without any query parameters

RN: This would imply a partial update of the resource 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?

4.5 **DELETE** resource instance

The HTTP DELETE method destroys a resource instance and any OCCI Links associated with the resource instance.

4.5.1 Client DELETE request

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

4.5.2 Server DELETE response

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

5 HTTP methods applied to collections URLs

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 resource 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 resource instances of that particular Kind.
- Mixin locations The location of an OCCI Mixin instance represents the collection of all resource 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 resource instances available.

5.1 GET collection

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

5.1.1 Client GET request

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

```
- ""
query-string
                    | "?" query-parameter *( "&" query-parameter )
  query-parameter
                   = attribute-filter
                    | category-filter
                    | pagination-start
                    | pagination-count
  attribute-filter = "q=" attribute-search *( "+" attribute-search )
  attribute-search = 1*( string-urlencoded |
                         attribute-name "%3D" string-urlencoded )
  category-filter = "category=" string-urlencoded
 pagination-start = "start=" 1*( DIGIT )
 pagination-count = "count=" 1*( DIGIT )
  attribute-name = attr-component *( "." attr-component )
                   = LOALPHA *( LOALPHA | DIGIT | "-" | "_" )
  attr-component
  string-urlencoded = *( ALPHA | DIGIT | "-" | "_" | "." | "~" | "%" )
```

Filtering A search filter can be applied to categories and attributes of resource 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 resource 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 resource instances of the Compute type: category=http%3A%2F%2Fschemas.ogf.org%2Focci%2Finfrastructure%23compute

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

The *start* parameter specifies the offset into the collection. A value of zero, *start=0* indicates the beginning of the collection. The *count* parameter sets the maximum number of elements to include in the response. For example ?start=20&count=10 would indicate the third page with a limit of 10 elements per page.

Example request

```
GET /storage/?q=ubuntu+server&start=0&count=20 HTTP/1.1
Host: example.com
Accept: application/occi+json
User-Agent: occi-client/x.x OCCI/1.1
```

5.1.2 Server GET response

```
HTTP/1.1 200 OK
Server: occi-server/x.x OCCI/1.1
Content-Type: application/occi+json; charset=utf-8
{
  "start": 0,
  "count": 3,
  "collection": [
    {
      "kind": { ... },
      "mixins": [ ... ],
      "actions": [ ... ],
      "links": [ ... ],
      "attributes": { ... },
      "location": "http://example.com/storage/e3578467-b6ba-448b-8032-5203278d54db"
   },
    { ... },
    { ... }
 ]
}
```

5.2 POST collection

The HTTP POST method is used to create/update one or more resource instances in a single atomic request. An OCCI server MUST identify existing resource instances using the occi.core.id attribute.

5.2.1 Client POST request

```
POST /storage/ HTTP/1.1
Host: example.com
Accept: application/occi+json
User-Agent: occi-client/x.x OCCI/1.1
Content-Type: application/occi+json; charset=utf-8
{
  "collection": [
    {
      "kind": { ... },
      "mixins": [ ... ],
      "links": [ ... ],
      "attributes": { ... },
   },
    { ... },
    { ... }
  ]
}
```

5.2.2 Server POST response

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

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

5.3 **POST** collection with "action" query parameter

todo

5.4 PUT collection

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

5.5 DELETE collection

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

6 More examples

The OCCI demo instance of occi-py⁶ running at http://www.nyren.net/api/ has an early version of the draft JSON rendering available. Feel free to play around with it. However, please note the following limitations:

- It does not support request data in JSON.
- Filtering and pagination is not yet supported.

A few example queries using curl:

```
curl -i -H 'accept: application/occi+json' http://www.nyren.net/api/-/
curl -i -H 'accept: application/occi+json' http://www.nyren.net/api/link/
curl -i -X POST -H 'accept: application/occi+json' http://www.nyren.net/api/compute/
```

⁶http://github.com/nyren/occi-py