GWD-R
OCCI-WG

Ralf Nyrén

February 25, 2011
Updated: February 27, 2012

Open Cloud Computing Interface - JSON Rendering

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.

Comments

Contents

1 Introduction

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].

The following terms [2] are used when referring to URI components:

```
 http://example.com:8080/over/there?action=stop#xyz
 \__/   \______________/\_________/ \_________/ \_/
  |           |              |            |       |
scheme     authority        path        query   fragment
```

3 OCCI JSON Rendering

TBD: intro, JSON Rendering is RESTful, can happily co-exist with the existing HTTP Rendering, etc

4 Namespace

The JSON Rendering maps the OCCI Core model into the URL hierarchy by binding Kind and Mixin instances to unique URL paths. Such an URL path is called the *location* of the Kind or Mixin. A provider is free to choose the *location* as long as it is unique within the service provider's URL namespace. For example, the Kind instance[1] 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 applies to the Kind instance for OCCI Entity.

4.1 Bound and unbound paths

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.

An unbound URL path MAY represent the union of all Kind and Mixin collection "below" the unbound path. RN: FIXME: Should this be a MUST instead?

4.2 Unique identifiers

In the OCCI JSON Rendering the `occi.core.id` attribute MUST be a UUID [3] in canonical form, for example `30782156-c5e1-43c2-99fd-7cf46bc2daa7`.

A resource instance MUST exist in the collection of its associated Kind and Mixins. Concatenation of a resource instance's Kind or Mixin location together with its `occi.core.id` MUST yield a valid URL from which the resource instance can be retrieved.

Example 1 A resource instance of the Network type has been assigned a unique identifier so that `occi.core.id` is `30782156-c5e1-43c2-99fd-7cf46bc2daa7`. The Kind of the Network type is bound to location `/network/`.

A request to `/network/30782156-c5e1-43c2-99fd-7cf46bc2daa7` MUST then yield a representation of the Network instance.

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

Example 2 The same resource instance as in example 1 is also associated with the IPNetwork Mixin which is bound to location /ipnetwork/.

A request to /ipnetwork/30782156-c5e1-43c2-99fd-7cf46bc2daa7 MUST yield a representation of the same Network instance as in example 1.

5 JSON format

The OCCI JSON Rendering uses four different formats to render the data exchanged between an OCCI client and server. Each data format has been assigned a separate media-type as summarised below. All four formats use a top-level JSON object with a different set of object members. The details of these structures are described in the subsections below.

application/occi-entity+json A single OCCI Resource or OCCI Link instance. See section 5.1.

application/occi-collection+json A collection of OCCI Resource and OCCI Link instances. See section 5.2.

application/occi-action+json Invocation of an OCCI Action. See section 5.3.

application/occi-discovery+json Format used by the OCCI discovery mechanism which comprise the OCCI type system [4]. See section 5.4.

An OCCI client MUST specify the media-type whenever request payload is submitted. Likewise an OCCI server MUST specify the media-type of the response payload. The media-type MUST be specified in the HTTP Content-Type header as specified in RFC2616 [5].

5.1 Single resource instance format

A single resource instance is rendered in the application/occi-entity+json format. A *resource instance* is a sub-type of OCCI Entity which has been instantiated. OCCI Resource and OCCI Link are the top-most sub-types of OCCI Entity. OCCI Entity itself cannot be instantiated.

The single resource instance format consists of a JSON object as shown in the following example. Table 1 define the object members.

```
{
  "kind": "http://schemas.ogf.org/occi/infrastructure#compute",
  "mixins": [ "http://example.com/templates/os#ubuntu-12.04" ],
  "actions": [
    {
      "title": "Start My Server",
      "href": "/compute/1154cb39-ea39-4a3a-a9da-c480968252f4?action=start",
      "rel": "http://schemas.ogf.org/occi/infrastructure/compute/action#start"
    }
  ],
  "links": [
    {
      "title": "My disk",
      "href": "http://example.com/storage/ff0fcfee-711f-4f33-8202-f0beac5427a6",
      "rel": [ "http://schemas.ogf.org/occi/infrastructure#storage" ],
      "link_href": "http://example.com/link/storage/17aef983-945a-4adf-a841-34abd7445fbb",
      "link_rel": [ "http://schemas.ogf.org/occi/infrastructure#storagelink" ],
      "attributes": { "occi.storagelink.deviceid": "ide:0:1" }
    }
  ],
  "attributes": {
    "occi.core.id": "1154cb39-ea39-4a3a-a9da-c480968252f4",
    "occi.compute.architecture": "x86_64",
    "occi.compute.hostname": "My Server",
    "occi.compute.speed": 2.67,
    "occi.compute.memory": 1.0,
    "occi.compute.state": "inactive"
  }
}
```

Table 1. A single resource instance is rendered using the the `application/occi-entity+json` format which consists of a JSON object with the members described in the table.

Object member	JSON type	Description	Applicable to
kind	string	Kind identifier	Request/response
mixin	array of strings	List of Mixin identifiers	Request/response
actions	array of objects	Actions applicable to the resource instance	Response *only*
links	array of objects	Associated OCCI Links	Request/response. OCCI Resource *only*
attributes	object	Instance attributes	Request/response

5.2 Collection format

A collection of resource instances is rendered in the `application/occi-collection+json` format. A collection may consist of resource instances of different types.

Every Kind and Mixin instance represent a collection of its associated resource instances. For example the Kind instance for the Compute type, `http://schemas.ogf.org/occi/infrastructure#compute`, automatically represent a collection of all Compute instances [4]. The collection format MAY be used to render a union of several collections as one big combined collection.

The collection format consists of a JSON object as shown in the example below. Table 2 define the object members.

```
{
  collection: [ ... ],
  size: 1004,
  limit: 50,
  next: "http://example.com/compute/?marker=25c46a34-3caa-4713-806b-6abb9593ba2d&limit=50"
}
```

Table 2. A collection of resource instances is rendered in the `application/occi-collection+json` format. The format consists of a JSON object with the following members.

Object member	JSON type	Description	Applicable to
collection	array of objects	List of resource instances	Request/response
size	integer	Total size of the collection	Response *only*
limit	integer	Effective page size limit	Response *only*
next	string	URL of the next page	Response *only*

The `collection` array consists of JSON objects in *single resource instance format* described in section 5.1.

The collection format has built-in support for pagination. Pagination allows a subset of a collection to be transferred to reduce server load and client processing. An OCCI server MUST specify the `next` member whenever a paginated collection is returned. See section 7.1.1.

5.3 Action invocation format

An action invocation is rendered in the `application/occi-action+json` format. An action is an invokable operation on a resource instance and is identified by a OCCI Category instance.

The action invocation format consists of a JSON object as shown in the example below. Table 3 define the object members. Action invocations are only relevant to client requests.

```
{
  "category": "http://schemas.ogf.org/occi/infrastructure/compute/action#stop",
  "attributes": {
    "method": "graceful"
  }
}
```

Table 3. An action represent an invocable operation on a resource instance. The `application/occi-action+json` format is used to render an action invocation. The format consists of a JSON object with the following members.

Object member	JSON type	Description
category	string	Category identifier
attributes	object	Action attributes

5.4 Discovery interface format

The OCCI discovery mechanism enables an OCCI client to discovery the types available for resource instantiation [4]. The OCCI JSON Rendering provides an interface to this mechanism called the *discovery interface*[2]. An OCCI client MAY add or remove custom Mixins through the discovery interface.

The discovery interface is rendered in the `application/occi-discovery+json` format. The same format is used for both request and response. The format consists of a JSON object as shown in the example below. Table 4 define the top-level object members. The object members describing a Kind or a Mixin are defined in table 5 while the object members describing a Category[3] are defined in table 6. The object members describing attribute properties are defined in table 7.

Table 4. The `application/occi-discovery+json` format is used to render the discovery mechanism. The format consists of a JSON object with the following members.

Object member	JSON type	Description	Applicable to
kinds	array of objects	List of Kind instances, see table 5.	Response *only*
mixins	array of objects	List of Mixin instances, see table 5.	Request/response
categories	array of objects	List of Category instances, see table 6.	Response *only*

```
{
  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": {
```

[2]Referred to as the "Query interface" in the OCCI HTTP Rendering [6].

[3]A Category instance is the identifier of an OCCI Action [4].

Table 5. Table 4 describe that Kind and Mixin instances are rendered as two separate arrays of JSON objects. Each such object has the following members.

Object member	JSON type	Description	Applicable to
term	string	Unique identifier within the categorisation scheme	Request/response
scheme	string	Categorisation scheme	Request/response
title	string	Display name	Request/response
related	string	Identifier of related “parent” Kind or Mixin instance	Request/response
attributes	object of name-object pairs	Attribute properties, see table 7	Request/response
actions	array of strings	List of action identifiers	Response *only*
location	string	URL bound to the Kind or Mixin instance	Request/response

Table 6. A Category instance is identify an action and are only relevant to responses from the discovery interface. Table 4 describe that Category instances are rendered as an array of JSON objects. Each such object has the following members.

Object member	JSON type	Description
term	string	Unique identifier within the categorisation scheme
scheme	string	Categorisation scheme
title	string	Display name
attributes	object	Attribute properties, see table 7

Table 7. Table 5 and 6 define the attribute properties of a Kind, Mixin or Category to be rendered as a JSON object consisting of attribute-name and properties-object pairs. The attribute-properties object has the members defined in this table. All attribute properties are optional and the table shows which property default value an OCCI client MUST assume if a particular property is unspecified. Only the `default` property is valid in client requests to the discovery interface. RN: Should we have both float and integer?

Object member	JSON type	Description	Default
mutable	boolean	Attribute mutability	true
required	boolean	Whether the attribute MUST be specified at resource instantiation	false
type	string	Enum {string, integer, float, boolean}	string
range	2 elements array	Lower and upper bound of an integer/float attribute	-
default	string, number or boolean	Attribute default when not specified by client. Implies required = false	-

```
          "mutable": true,
          "required": false,
          "type": "string",
          "default": "x86_64"
        },
        "occi.compute.cores": {
          "mutable": true,
          "required": false,
          "type": "integer",
          "range": [ 1, 24 ],
          "default": 1
        },
        ...
      }
      "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": "/compute/"
    },
    ...
```

```
  ],
  mixins: [
    {
      "term": "medium",
      "scheme": "http://example.com/templates/resource#",
      "title": "Medium VM",
      "related": "http://schemas.ogf.org/occi/infrastructure\#resource\_tpl",
      "attributes": {
        "occi.compute.cores": { "default": 4 },
        "occi.compute.memory": { "default": 8.0 },
        "occi.compute.speed": { "default": 2.8 }
      }
    },
    ...
  ],
  categories: [
    {
      "term": "start",
      "scheme": "http://schemas.ogf.org/occi/infrastructure/compute/action#",
      "title": "Start Compute Resource",
      "attributes": {
        "method": { "mutable": true, "required": false, "type": "string" }
      }
    },
    ...
  ]
}
```

6 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 server MUST use the request processing defined by this document whenever the Content-Type of a request use one of the media-types defined in section 5. In order to receive a response defined by the OCCI JSON Rendering a client MUST include one or several of the JSON media-types[4] in the Accept header.

At the beginning of every "Client request" and "Server response" sub-section below the required Accept and Content-Type headers are defined. A client MAY supply all four JSON media-types in the Accept header in which case the server MUST choose the most suitable format.

6.1 GET resource instance

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

6.1.1 Client GET request

Accept `application/occi-entity+json`

The body of the HTTP GET request MUST be empty.

[4]See section 5.

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

6.1.2 Server GET response

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

The body of the HTTP GET response MUST contain a rendering of the resource instance in the single-resource-instance format, see section 5.1.

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

{
  "kind": "http://schemas.ogf.org/occi/infrastructure#compute",
  "mixins": [ ... ],
  "actions": [ ... ],
  "links": [ ... ],
  "attributes": { ... }
}
```

6.2 PUT resource instance

The HTTP PUT method creates or replaces the resource instance at the specified URL. Since the UUID of the resource instance is supplied in the request URL an OCCI server MAY refuse to create a new instance. If specified the `occi.core.id` attribute MUST be identical to the UUID in the request URL.

6.2.1 Client PUT request

Accept `application/occi-entity+json`

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

The full representation of the resource instance MUST be supplied in the HTTP body of the request. The request body MUST use the single-resource-instance format defined in section 5.1.

The object member `links` MUST NOT be supplied by a client. Removing a set of OCCI Links breaks the idempotence of a PUT request. Any Links present for an existing OCCI Resource MUST be left intact. A server MUST refuse a request containing the `links` member.

```
PUT /compute/012d2b48-c334-47f2-9368-557e75249042 HTTP/1.1
Host: example.com
Accept: application/occi-entity+json
User-Agent: occi-client/x.x OCCI/1.1
Content-Type: application/occi-entity+json; charset=utf-8

{
  "kind": "http://schemas.ogf.org/occi/infrastructure#compute",
  "mixins": [ ... ],
  "attributes": { ... }
}
```

6.2.2 Server PUT response

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 resource instance in single-resource-instance format. The response MUST be identical[5] to that of a subsequent GET request of the same URL.

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

{
  "kind": "http://schemas.ogf.org/occi/infrastructure#compute",
  "mixins": [ ... ],
  "actions": [ ... ],
  "links": [ ... ],
  "attributes": { ... }
}
```

6.3 POST resource instance (action)

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 5.3. 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.

6.3.1 Client POST action request

Accept `application/occi-entity+json`

Content-Type `application/occi-action+json`

The example shows the combined method.

```
POST /compute/012d2b48-c334-47f2-9368-557e75249042?action=stop HTTP/1.1
Host: example.com
Accept: application/occi-entity+json
User-Agent: occi-client/x.x OCCI/1.1
Content-Type: application/occi-action+json; charset=utf-8

{
  "category": "http://schemas.ogf.org/occi/infrastructure/compute/action#stop",
  "attributes": {
    "method": "graceful"
  }
}
```

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

6.3.2 Server POST action response

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 resource instance. Otherwise the server MUST return status code 204 and no response payload.

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

6.4 POST resource instance

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

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?

6.5 DELETE resource instance

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

6.5.1 Client DELETE request

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

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

6.5.2 Server DELETE response

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

7 HTTP methods applied to collections URLs

RN: 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 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.

7.1 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.

7.1.1 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-marker
                    | pagination-limit
  attribute-filter  = "q=" attribute-search *( "+" attribute-search )
  attribute-search  = 1*( string-urlencoded |
                          attribute-name "%3D" string-urlencoded )
  category-filter   = "category=" string-urlencoded
  pagination-marker = "marker=" UUID
  pagination-limit  = "limit=" 1*( DIGIT )
  attribute-name    = attr-component *( "." attr-component )
  attr-component    = LOALPHA *( LOALPHA | DIGIT | "-" | "_" )
  string-urlencoded = *( ALPHA | DIGIT | "-" | "_" | "." | "~" | "%" )
```

RN: FIXME: UUID in ABNF

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 RN: 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.

Example request

```
GET /storage/?q=ubuntu+server&limit=20 HTTP/1.1
Host: example.com
Accept: application/occi-collection+json
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

{
  "collection": [
    {
      "kind": "..." ,
      "mixins": [ ... ],
      "actions": [ ... ],
      "links": [ ... ],
      "attributes": { ... },
    },
    { ... },
    { ... }
  ],
  "limit": 20,
  "size": 137,
  "next": "http://example.com/storage/?q=ubuntu+server&marker=59b50...9b3&limit=20"
}
```

7.2 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.

7.2.1 Client POST request

```
POST /storage/ HTTP/1.1
Host: example.com
Accept: application/occi-collection+json
User-Agent: occi-client/x.x OCCI/1.1
Content-Type: application/occi-collection+json; charset=utf-8

{
  "collection": [
    {
      "kind": "...",
      "mixins": [ ... ],
      "links": [ ... ],
      "attributes": { ... },
    },
    { ... },
    { ... }
  ]
}
```

7.2.2 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?

7.3 POST collection with “action” query parameter

todo

7.4 PUT collection

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

7.5 DELETE collection

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

8 Glossary

Term	Description
Action	An OCCI base type. Represent an invocable operation on a Entity sub-type instance or collection thereof.
Category	A type in the OCCI model. The parent type of Kind.
Client	An OCCI client.
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.
Kind	A type in the OCCI model. A core component of the OCCI classification system.
Link	An OCCI base type. A Link instance associate one Resource instance with another.
mixin	An instance of the Mixin type associated with a **resource instance**. The “mixin” concept as used by OCCI *only* applies to instances, never to Entity types.
Mixin	A type in the OCCI model. A core component of the OCCI classification system.
OCCI	Open Cloud Computing Interface.
OCCI base type	One of Entity, Resource, Link or Action.
OGF	Open Grid Forum.
Resource	An OCCI base type. The parent type for all domain-specific resource types.
resource instance	An instance of a sub-type of Entity. The OCCI model defines two sub-types of Entity, the Resource type and the Link type. However, the term *resource instance* is defined to include any instance of a *sub-type* of Resource or Link as well.
Tag	A Mixin instance with no attributes or actions defined.
Template	A Mixin instance which if associated at resource instantiation time pre-populate certain attributes.
type	One of the types defined by the OCCI model. The OCCI model types are Category, 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.

9 Intellectual Property Statement

10 Disclaimer

11 Full Copyright Notice

References

[1] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119 (Best Current Practice), Internet Engineering Task Force, Mar. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2119.txt

[2] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986 (Standard), Internet Engineering Task Force, Jan. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc3986.txt

[3] P. Leach, M. Mealling, and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace," RFC 4122 (Proposed Standard), Internet Engineering Task Force, Jul. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4122.txt

[4] R. Nyrén, A. Edmonds, A. Papaspyrou, and T. Metsch, "Open Cloud Computing Interface – Core," GFD-P-R.183, April 2011. [Online]. Available: http://ogf.org/documents/GFD.183.pdf

[5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616 (Draft Standard), Internet Engineering Task Force, Jun. 1999, updated by RFCs 2817, 5785. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

[6] T. Metsch and A. Edmonds, "Open Cloud Computing Interface – HTTP Rendering," GFD-P-R.185, April 2011. [Online]. Available: http://ogf.org/documents/GFD.185.pdf