TBD ...Clinton DeWittRobert
SyreMark NottinghamIan
HicksonDavid HyattLeonard
RichardsonSam RubyMark
NottinghamSam Johnston, TBD
Google, Inc. Apple, Inc.
September 16, 2009

Open Cloud Computing Interface Specification

Status of This Document

TBD...

Abstract

TBD...

Contents

TOC comes here

1. Introduction

An overview of the document.

2. OCCI Walkthrough

2.1. Overview

This document may lag behind the actual specification

The Open Cloud Computing Interface (OCCI) is an API for managing cloud infrastructure services (also known as Infrastructure as a Service or IaaS) which strictly adheres to REpresentational State Transfer (REST) principles and is closely tied to HyperText Tranfer Protocol (HTTP). For simplicity and scalability reasons it specifically avoids Remote Procedure Call (RPC) style interfaces and can essentially be implemented as a horizontally scalable document repository with which both nodes and clients interact.

This document describes a step-by-step walkthrough of performing various tasks as at the time of writing.

2.2. Getting Started

Connecting

Each implementation has a single OCCI end-point URL (we'll use http://example.com/) and everything you need to know is linked from this point - configuring clients is just a case of providing this parameter. In the simplest case the end-point may contain only a single resource or type of resource (e.g. a hypervisor burnt into the BIOS of a motherboard exposing compute resources, a network switch/router exposing network resources or a SAN exposing storage resources) and at the other end of the spectrum it may provide access to a global cloud infrastructure (e.g. the "Great Global Grid" or GGG). You will only ever see those resources to which you have access to (typically all of them for a private cloud or a small subset for a public cloud) and flexible categorisation and search provide fine-grained control which resources are returned, allowing OCCI to handle the largest of installations. You will always connect to this end-point over HTTP(S) and given the simplicity of the interface most user-agents are suitable, including libraries (e.g. urllib2, LWP), command line tools (e.g. curl, wget) and full blown browsers (e.g. Firefox).

Authenticating

When you connect you will normally be challenged to authenticate via HTTP (this is not always the case - in secure/offline environments it may not be necessary) and will need to do

so via the specified mechanism. It is anticipated that most implementations will require HTTP Basic Authentication over SSL/TLS so at the very least you should support this (fortunately almost all user-agents already do), but more advanced mechanisms such as NTLM or Kerberos may be deployed. Certain types of accesses (such as a compute resource querying OCCI for introspection and configuration) may be possible anonymously (having already been authenticated by interface and/or IP address). Should you be redirected by the API to a node, storage device, etc. (for example, to retrieve a large binary representation) then you should either be able to transparently authenticate or a signed URL should be provided. That is, a single set of credentials is all that is required to access the entire system from any point.

Representations

As the resource itself (e.g. a physical machine, storage array or network switch) cannot be transferred over HTTP (at least not yet!) we instead make available one or more representations of that resource. For example, an API modeling a person might return a picture, fingerprints, identity document(s) or even a digitised DNA sequence, but not the person themselves. A circle might be represented by SVG drawing primatives or any three distinct points on the curve. For cloud infrastructure there are many useful representations, and while OCCI standardises a number of them for interoperability purposes, an implementation is free to implement others in order to best serve the specific needs of their users and to differentiate from other offerings. Other examples include:

- Open Cloud Computing Interface (OCCI) descriptor format (application/occi+xml)
- Open Virtualisation Format (OVF) file (application/ovf+xml?)
- Open Virtualisation Archive (OVA) file (application/x-ova?)
- Screenshot of the console (image/png)
- Access to the console (application/x-vnc)

The client indicates which representation(s) it desires by way of the URL and/or HTTP Accept headers (e.g. HTTP Content Negotiation) and if the server is unable to satisfy the request then it should return HTTP 406 Not Acceptable.

Descriptors

In addition to the protocol itself, OCCI defines a simple key/value based descriptor format for cloud infrastructure resources:

compute	Provides computational services, ranging from dedicated physical machines (e.g. Dedibox) to virtual machines (e.g. Amazon EC2) to slices/zones/containers (e.g. Mosso Cloud Servers).
network	Provides connectivity between machines and the outside world. Usually virtual and may or may not be connected to a physical segment.
storage	Provides storage services, typically via magnetic mass storage devices (e.g. hard drives, RAID arrays, SANs).

Given the simplicity of the format it is trivial to translate between wire formats including plain text, JSON, XML and others. For example:

```
occi.compute.cores 2
compute.speed 3200
compute.memory 2048
```

TBD ...Clinton DeWittRobert SyreMark NottinghamIan HicksonDavid HyattLeonard RichardsonSam RubyMark NottinghamSam Johnston, TBD
Google, Inc. Apple, Inc.

Each resource is identified by its dereferenceable URL which is by definition unique, giving information about the origin and type of the resource as well as a local identifier (the combination of which forms a globally unique compound key). The primary drawback is that the more information that goes into the key (and therefore the more transparent it is), the more likely it is to change. For example, if you migrate a resource from one implementation to another then its identifier will change (though in this instance the source should provide a HTTP 301 Moved Permanently response along with the new location, assuming it is known, or HTTP 410 Gone otherwise).

In order to realise the benefit of transparent, dereferenceable identifiers while still being able to track resources through their entire lifecycle an immutable UUID attribute should be allocated which will remain with the resource throughout its life. This is particularly important where the same resource (e.g. a network) appears in multiple places.

New implementations should use type 4 (random) UUIDs anyway, as these can be safely allocated by any node without consulting a register/sequence, but where existing identifiers are available they should be used instead (e.g. http://amazon.com/compute/ami-ef48af86).

2.3. Operations

Create

To create a resource simply POST it to the appropropriate collection (e.g. /compute, /network or /storage) as an HTML form (supported by virtually all user agents) or in another supported format (e.g. OVF):

```
POST /compute HTTP/1.1
Host: example.com
Content-Length: 35
Content-Type: application/x-www-form-urlencoded

compute.cores=2&compute.memory=2048
```

Rather than generating the new resource from scratch you may also be given the option to GET a template and POST or PUT it back (for example, where "small", "medium" and "large" instances or pre-configured appliances are offered).

Retrieve

The simplest command is to retrieve a single resource by conducting a HTTP GET on its URL (which doubles as its identifier):

```
GET /compute/b10fa926-41a6-4125-ae94-bfad2670ca87 HTTP/1.1
Host: example.com
```

This will return *a HTTP 300 Multiple Choices response containing a list of available representations for the resource as well as a suggestion in the form of a HTTP Location: header of* the default rendering, which should be HTML (thereby allowing standard browsers to access the API directly). An arbitrary number of alternatives may also be returned by way of HTTP Link: headers.

If you just need to know what representations are available you should make a HEAD request instead of a GET - this will return the metadata in the headers without the default rendering.

Some requests (such as searches) will need to return a collection of resources. There are two options:

Pass-by-reference	A plain text or HTML list of links is provided but each needs to be retrieved separately, resulting in O(n+1) performance.

TBD ...Clinton DeWittRobert SyreMark NottinghamIan HicksonDavid HyattLeonard RichardsonSam RubyMark NottinghamSam Johnston, TBD
Google, Inc.Apple, Inc.

Pass-by-value	A wrapper format such as Atom is used to deliver [link to] the content as well as the metadata (e.g. links, associations, cahching information, etc.), resulting in O(1) performance.

Update

Updating resources is trivial - simply GET the resource, modify it as necessary and PUT it back where you found it.

Delete

Simply `DELETE` the resource:

```
DELETE /compute/b10fa926-41a6-4125-ae94-bfad2670ca87 HTTP/1.1
Host: example.com
```

2.4. Sub-resource Collections

(For want of a better name)

Each resource may expose collections for functions such as logging, auditing, change control, documentation and other operations (e.g. http://example.com/compute/123/log/456) in addition to any required by OCCI. As usual CRUD operations map to HTTP verbs (as above) and clients can either PUT entries directly if they know or will generate the identifiers, or POST them to the collection if this will be handled on the server side (using POST Once Exactly (POE) to ensure idempotency).

Requests

Requests are used to trigger state changes and other operations such as backups, snapshots, migrations and invasive reconfigurations (such as storage resource resizing). Those that do not complete immediately (returning HTTP 200 OK or similar) must be handled asynchronously (returning HTTP 201 Accepted or similar).

```
POST /compute/123/requests HTTP/1.1
Host: example.com
Content-Length: 35
Content-Type: application/x-www-form-urlencoded

state=shutdown&type=acpioff
```

The actual operation may not start immediately (for example, backups which are only handled daily at midnight) and may take some time to complete (for example a secure erase which requires multiple passes over the disk). Clients can poll for status periodically or use server push (or a non-HTTP technology such as XMPP) to monitor for events.

3. OCCI Frequently Asked Questions

3.1. General

Who created the Open Cloud Computing Interface (OCCI)?	The Open Grid Forum (OGF)'s Open Cloud Computing Interface Working Group (OCCI-WG) created the Open Cloud Computing Interface (OCCI).
Who are the Open Cloud Computing Interface Working Group (OCCI-WG) officials?	Andrew Edmonds (Intel, SLA@SOI), Thijs Metsch (Sun Microsystems) and Alexis Richardson (Rabbit Technologies Ltd) are the chairs, and Sam Johnston (Australian Online Solutions) is the secretary.
Who else was involved?	Around 200 individuals representing over 100 companies were involved in the development of the Open Cloud Computing Interface (OCCI).

3.2. Use Cases

How were the use cases collected?	Use cases were solicited from the working group mailing list as well as other external sources.

3.3. Technical

Why didn't you invent your own XML representation?	See Tim Bray's Don't Invent XML Languages post.

4. OCCI Core Specification

4.1. Introduction

The Open Cloud Computing Interface is an open community consensus API, initially targeting cloud infrastructure services or "Infrastructure as a Service (IaaS)". A "Resource Oriented Architecture (ROA)", it is as close as possible to the underlying HyperText Transfer Protocol (HTTP), deviating only where absolutely necessary. Each resource (identified by a canonical URL) has zero or more representations which may or may not be hypertext (e.g. HTML). Metadata including associations between resources is exposed via HTTP headers (e.g. the Link: header), except in the case of collections where Atom is used as the meta-model.

Tip

Some resources can be interacted with but not rendered due to the nature of the resource or prevailing security policies.

4.2. Basics

URL Namespace

The interface is defined by a single URL entry point which will either be a *collection*, contain *link*(s) to *collection*(s) (*in-band* and/or *out-of-band*) or both.

Kinds, Actions and Attributes

An interface exposes "kinds" which have "attributes" and on which "actions" can be performed. The attributes are exposed as key-value pairs and applicable actions as links, following HATEOAS principles (whereby state transitions are defined *in-band* rather than via rules).

CRUD Operations

Create, Retrieve, Update and Delete (CRUD) operations map to the POST, GET, PUT and DELETE HTTP verbs respectively. HEAD and OPTIONS verbs may be used to retrieve metadata and valid operations without the entity body to improve performance. WebDAV definitions are used for MOVE and COPY. All existing HTTP features is available for caching, proxying, gatewaying and other advanced functionality.

POST (Create)	"The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line."RFC2616 POSTing a representation (e.g. OVF) to a collection (e.g. /compute) will result in a new resource being created (e.g. /compute/123) and returned in the Location: header. POST is also used with HTML form data to trigger verbs (e.g. restart)

GET (Retrieve - Metadata and Entity)	“The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI.”RFC2616 GETting a resource (e.g. /compute/123) will return a representation of that resource in the most appropriate supported format specified by the client in the Accept header. Otherwise "406 Not Acceptable" will be returned.
PUT (Create or Update)	“The PUT method requests that the enclosed entity be stored under the supplied Request-URI.”RFC2616 PUTting a representation (e.g. OVF) to a URL (e.g. /compute/123) will result in the resource being created or updated. The URL is known or selected by the client (in which case UUIDs should be used), in contrast to POSTs where the URL is selected by the server.
DELETE (Delete)	“The DELETE method requests that the origin server delete the resource identified by the Request-URI.”RFC2616 DELETE results in the deletion of the resource (and everything "under" it, as appropriate).

Additionally the following HTTP methods are used:

COPY (Duplicate)	“The COPY method creates a duplicate of the source resource identified by the Request-URI, in the destination resource identified by the URI in the Destination header.”RFC4918
HEAD (Retrieve - Metadata Only)	“The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response.”RFC2616
MOVE (Relocate)	“The MOVE operation on a non-collection resource is the logical equivalent of a copy (COPY), followed by consistency maintenance processing, followed by a delete of the source, where all three actions are performed in a single operation.”RFC4918
OPTIONS	“The OPTIONS method represents a request for information about the communication options available on the request/response chain identified by the Request-URI.”RFC2616

4.3. Connection

4.3.1. Authentication

Servers *may* require that requests be authenticated using standard HTTP-based authentication mechanisms (including OAuth).OAuth They indicate this requirement by returning `HTTP 401` with a `WWW-Authenticate` header and a suitable challenge (e.g. `Basic`, `Digest`, `OAuth`). The client then includes appropriate `Authorization` headers in its responses.RFC2617

Servers *may* set and clients *may* accept *cookies* in order to maintain authentication state between requests. Such sessions *should not* be used for other purposes in line with RESTful principles.RFC2109
TODO: Add support for SAML 2?

TBD ...Clinton DeWittRobert SyreMark NottinghamIan HicksonDavid HyattLeonard RichardsonSam RubyMark NottinghamSam Johnston, TBD Google, Inc.Ajoy Apple, Inc.Scott Morrison

4.3.2. Versioning

Every request *should* include an `OCCI-Version` HTTP header indicating the version of the API requested (e.g. `1.0`). If none is provided the latest available version *shall* be used.

4.4. Model

The model defines the objects themselves without regard to how they interrelate.

4.4.1. Kinds

Each category of resources distinguished by some common characteristic or quality is called a *kind* (e.g. `compute`, `network`, `storage`, `queue`, `application`, `contact`).

Kinds defined by this standard live in the `http://purl.org/occi/kind/` namespace but anyone can define a new kind by allocating a URI they control.

Warning

Defining your own kinds can lead to interoperability problems and should be a last resort reserved for unique functionality. A simple peer review process is available for extending the registries which should be used where possible.

Each resource *must* specify a kind by way of a *category* within the *scheme* "`http://purl.org/occi/kind/`".

Tip

The word *type* is not used in this context in order to avoid confusion with Internet media types.

4.4.2. Attributes

An *attribute* is a specification that defines a property of an object. It is expressed in the form of key-value pairs.

Attributes are divided into namespaces which are separated by the dash character ("-"). They *must* be handled as case-insensitive but *should* be case-preserving by default (depending on the format).

Tip

This scalable approach was derived from the Mozilla Firefox `about:config` page, though the "." separator was replaced with "-" for maximum compatibility with various formats.

Attributes defined by this standard reside under the `OCCI` namespace (e.g. "`OCCI-ABC`") but anyone can define a new attribute by allocating a unique namespace under "`X-`" (e.g. "`X-Acme-ABC`"). A number of attributes are common to all *kind*s.

Warning

Defining your own attributes can lead to interoperability problems and should be a last resort reserved for unique functionality. A simple peer review process is available for extending the registries which should be used where possible.

```
OCCI-Compute-Cores: 2
```

```
OCCI-Compute-Speed: 3000
OCCI-Memory-Size: 8192
Acme-Network-Identifier: dmz
```

Table 1. Common Attributes

Attribute	Description	Example
`OCCI-Id`	Immutable identifier for the resource	`urn:uuid:d0e9f0d0-f62d-4f28-bc90-23b0bd871770`
`OCCI-Kind`	Kind of resource	`compute`
`OCCI-Title`	Display name for the resource	`Compute Resource #123`
`OCCI-Summary`	Description of the resource	`A virtual compute resource`
`OCCI-Author-Name`	Owner of the resource	`John Citizen`
`OCCI-Updated`	Last updated date/time [RFC3339]	`2020-12-31T23:59:59Z`
`OCCI-Version`	Specification version	`1.0`
`ETag`	HTTP Entity Tags [RFC2616]	`"dad86c61eea237932f"`

4.4.3. Actions

An *action* is some process that can be carried out on one or more *resource*s.

Each available *action* for a given *resource* is indicated via a *link* with the `action` class.

```
<link rel="http://purl.org/occi/action/restart#cold"
      class="action"
      title="Cold Restart"
      href="http://example.com/123/restart;type=cold" />
```

Actions defined by this standard reside under the `http://purl.org/occi/action/` namespace but anyone can define a new action by allocating a URI they control.

Warning

Defining your own actions can lead to interoperability problems and should be a last resort reserved for unique functionality. A simple peer review process is available for extending the registries which should be used where possible.

An *action* is triggered via an HTTP POST and depending on the action requested (e.g. `resize`), parameters *may* be provided using HTML forms (e.g. `application/x-www-form-encoded`).

The specific parameters required and allowable values for them depend on the action and for advanced actions *may* require sending of custom *content type*s rather than `application/x-www-form-encoded`.

Synchronous actions *may* return `200 OK` on successful completion or `201 Created` with a `Location:` header indicating a new resource for audit purposes.

Tip

Assume that clients are paranoid and want audit trails for all but the most trivial of actions.

TBD ...Clinton DeWittRobert SyreMark NottinghamIan HicksonDavid HyattLeonard RichardsonSam RubyMark NottinghamSam Johnston TBD
Google, Inc. Apple, Inc.
September 16, 2009

In the event that the *action* does not complete immediately it should return HTTP 202 `Accepted` and a `Location:` header indicating a new resource where status and other pertinent information can be obtained.

> **Tip**
>
> Don't keep clients waiting - if you're not sure to return immediately then give them a resource they can monitor.

4.5. Meta-model

The meta-model defines how objects interrelate.

4.5.1. Categories

Category information allows for flexible organisation of resources into one or more vocabularies (each of which is referred to as a *scheme*).

The meta-model was derived from Atom, consisting of three attributes:

term	The term itself (e.g. “`compute`”)
scheme (optional)	The vocabulary (e.g. “`http://purl.org/occi/kind/`”)
label (optional)	A human-friendly display name for the term (e.g. “`Compute Resource`”)

```
<category term="compute"
          scheme="http://purl.org/occi/kind/"
          label="Compute Resource" />
```

Category schemes and/or terms defined by this standard reside throughout the `http://purl.org/occi/` namespace but anyone can define a new scheme by allocating a URI they control.

> **Tip**
>
> Categories provide a flexible way to manage resources by taxonomy (categories) and/or folksonomy (tags), where both can be shared between [groups of] users or globally. For example, users can create schemes for resource locations (e.g. `US-East`, `US-West`, `Europe`), operating systems (e.g. `Windows`, `Linux`) and patch levels (e.g.

TODO: Consider moving to link relations for categories so as to be compatible with existing standards rather than creating new ones. LINK is already standardised within HTML and HTTP and the Web Linking Internet-Draft will proceed to standard status. The Web Category draft is less sure, particularly where a workaround exists.

4.5.2. Collections

Where an operation returns multiple resources (e.g. categories, searches) this is referred to as a *collection*.

Depending on the format these are returned as:

- A list of pointers to resources (e.g. `text/uri-list`)
- A list of pointers to resources with metadata (e.g. `application/atom+xml` with `link` to content)

TBD ...Clinton DeWittRobert SyreMark NottinghamIan HicksonDavid HyattLeonard RichardsonSam RubyMark NottinghamSam Johnston, TBD
Google, Inc. Apple, Inc.

- A list of embedded resources and metadata (e.g. `application/atom+xml` with `content` embedded)

> **Tip**
>
> Most collections should be pointers to resources with metadata for performance reasons - O(1) rather than O(n+1) requests for pointers alone. The resources themselves should only be embedded when they are known to be of a reasonable size.

Any given URL can be a collection and/or advertise *link*s to other *collection*s using the `collection` class.

> **Tip**
>
> The root ("/") *should* expose collections *in-band* and/or *out-of-band* in order for clients to discover resources.

```
<link rel="http://purl.org/occi/collection/audit"
      class="collection"
      title="Audit Entries"
      href="http://example.com/123/audit" />
```

Paging

Collections *may* be divided into *page*s, with each linking to the "first", "last", "next" and "previous" *link relation*s.

```
<link rel="first" href="http://example.com/xyz;start=0" />
<link rel="previous" href="http://example.com/xyz;start=400" />
<link rel="self" href="http://example.com/xyz;start=500" />
<link rel="next" href="http://example.com/xyz;start=600" />
<link rel="last" href="http://example.com/xyz;start=900" />
```

4.5.3. Linking

Existing linking standards defined for Atom [RFC4287], HTTP [LINK] and HTML [HTML5] are used to indicate associations between resources. All formats *must* support *in-band* linking including:

- Link relations (e.g. `rel="alternate"`)
- Pointers to resources (e.g. `href="http://example.com/"`)
- Internet media types (e.g. `type="text/html"`)
- Extensibility (e.g. `attribute="value"`)

```
<link rel="related"
      title="System Documentation"
      href="http://example.com/user-guide.pdf"
      type="application/pdf" />
```

*Link relation*s defined by this standard reside under the `http://purl.org/occi/rel` namespace but anyone can define a new *link relation* by allocating a URI they control.

TBD ...Clinton DeWittRobert SyreMark NottinghamIan HicksonDavid HyattLeonard RichardsonSam RubyMark NottinghamSam Johnston, TBD
Google, Inc. Apple, Inc.

Table 2. Link Relations

Relation	Description
`category` (`http://purl.org/occi/rel#category`)	A category mapping whereby: • The `scheme` is required and indicated by the `href` attribute. • The `label` is optional and indicated by the `title` attribute. • The `term` is required and indicated by the `term` extended attribute.
`collection` (`http://purl.org/occi/rel#collection`)	A related collection whereby: • The root of the collection is indicated by the `href` attribute. • The *kind* of the collection is indicated by the `kind` extended attribute.
`first`	“An IRI that refers to the furthest preceding resource in a series of resources.” [LINK]
`help`	“The referenced document provides further help information for the page as a whole.” [HTML5]
`icon`	“The specified resource is an icon representing the page or site, and should be used by the user agent when representing the page in the user interface.” [HTML5]
`last`	“An IRI that refers to the furthest following resource in a series of resources.” [LINK]
`next`	“A URI that refers to the immediately following document in a series of documents.” [LINK]
`previous`	“A URI that refers to the immediately preceding document in a series of documents.” [LINK]
`search`	“The referenced document provides an interface specifically for searching the document and its related resources.” [HTML5, OpenSearch]
`self`	“Identifies a resource equivalent to the containing element” [RFC4287]

4.6. Extensibility

The interface is fully extensible, both via a public peer review process (in order to update the specification itself, usually via registries) and via independent allocation of unique namespaces (in order to cater for vendor-specific enhancements).

4.6.1. Foreign markup

Implementations *must* accept and forward but otherwise ignore markup they do not understand.

4.7. Security Considerations

Encryption is not required by the specification in order to cater for sites that do not or cannot use it (e.g. due to export restrictions, performance reasons, etc.), however SSL/TLS *should* be used over public networks including the Internet.

4.8. Registration

4.8.1. IANA Considerations

Internet Media Types (MIME Types)

The following media types are to be registered:

- text/occi
- application/occi+atom
- application/occi+json

Well-Known URI Registry

The following well-known URI suffix is to be registered:

URI Suffix	/.well-known/*occi*/
Change Controller	OGF
Specification Document	Open Cloud Computing Interface (OCCI) [`http://purl.org/occi`]
Related Information	N/A

Link Relation Type Registry

The following *link relation*s are to be registered:

- Category

Relation Name	category
Description	Assigns the link's context to a category, whereby the `scheme` is required and indicated by the `href` attribute, the `label` is optional and indicated by the `title` attribute and the `term` is required and indicated by the `term` extended attribute.
References	• Atom [RFC4287] • OCCI [this specification]
Notes	Category meta-model was derived from Atom for use with OCCI. This relation was defined for compatibility with existing standards including HTTP and HTML.

- Collection

Relation Name	collection
Description	Identifies a related *collection* whereby the root of the collection is indicated by the `href` attribute and the *kind* of the collection is indicated by the `kind` extended attribute.

Glossary

in-band	“Sending of metadata and control information in the same band, on the same channel, as used for data”, for example, by embedding it in HTML. [`http://en.wikipedia.org/wiki/In-band`]
kind	“A category of things distinguished by some common characteristic or quality”, for example events, messages, media. [`http://wordnetweb.princeton.edu/perl/webwn?s=kind`]
out-of-band	“Communications which occur outside of a previously established communications method or channel”, for example, in HTTP headers. [`http://en.wikipedia.org/wiki/Out-of-band_signaling`]
type	Internet media (MIME) type as defined by RFC2045 and RFC2046

Bibliography

Normative References

[RFC2045] *RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. `http://tools.ietf.org/html/rfc2045`. Internet Engineering Task Force (IETF) 1996-11.

[RFC2046] *RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. `http://tools.ietf.org/html/rfc2046`. Internet Engineering Task Force (IETF) 1996-11.

[RFC2109] *RFC 2109 - HTTP State Management Mechanism*. `http://tools.ietf.org/html/rfc2109`. Internet Engineering Task Force (IETF) 1997-02.

[RFC2616] *RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1*. `http://tools.ietf.org/html/rfc2616`. Internet Engineering Task Force (IETF) 1999-06.

[RFC2617] *RFC 2617 - HTTP Authentication: Basic and Digest Access Authentication*. `http://tools.ietf.org/html/rfc2617`. Internet Engineering Task Force (IETF) 1999-06.

[RFC3339] *RFC 3339 - Date and Time on the Internet: Timestamps*. `http://tools.ietf.org/html/rfc3339`. Internet Engineering Task Force (IETF) 2002-07.

[RFC4918] *RFC 4918 - HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)*. `http://tools.ietf.org/html/rfc4918`. Internet Engineering Task Force (IETF) 2007-06.

[OpenSearch] *OpenSearch 1.1*. `http://www.opensearch.org/Specifications/OpenSearch/1.1`. A9.com, Inc. (an Amazon company) Clinton DeWitt. Joel Tesler . Michael Fagan . Joe Gregorio . Aaron Sauve . James Snell . 2009.

Informative References

[RFC4287] *RFC 4287 - The Atom Syndication Format*. `http://tools.ietf.org/html/rfc4287`. Robert Syre. Mark Nottingham. Internet Engineering Task Force (IETF) 2005-12.

TBD ...Clinton DeWittRobert SyreMark NottinghamIan HicksonDavid HyattLeonard RichardsonSam RubyMark NottinghamSam Johnston, TBD
Google, Inc. Apple, Inc.
September 16, 2009

[HTML5] *HTML 5*. http://www.w3.org/TR/html5/. Ian Hickson. David Hyatt. World Wide Web Consortium (W3C) 2009-08-25.

[OAuth] *OAuth*. http://oauth.net/core/1.0. OAuth Core Workgroup <spec@oauth.net>. 2007-12-04.

[RWS] *RESTful Web Services*. http://oreilly.com/catalog/9780596529260/. 9780596529260. O'Reilly Media Leonard Richardson. Sam Ruby. 2007-05.

[LINK] *Web Linking*. http://tools.ietf.org/html/draft-nottingham-http-link-header. Internet Engineering Task Force (IETF) Mark Nottingham. 2009-07-12.

[CATEGORY] *Web Categories*. http://tools.ietf.org/html/draft-johnston-http-category-header. Internet Engineering Task Force (IETF) Sam Johnston. 2009-07-1.

5. OCCI Infrastructure

OCCI Infrastructure defines three kinds and various extensions relating to management of cloud infrastructure services (IaaS).

Table 3. Common Attributes

Attribute	Type	Description
OCCI-Infrastructure-Hostname	String	Valid DNS hostname for the resource (may be FQDN)

5.1. Kinds

Cloud infrastructure can be modeled using three primary kinds: compute, network and storage.

Table 4. Kinds

Kind	URI	Description
compute	http://purl.org/occi/kind/compute	Information processing resources
network	http://purl.org/occi/kind/network	Interconnection resources
storage	http://purl.org/occi/kind/storage	Recorded information resources

5.1.1. Compute

A compute resource is capable of conducting computations (e.g. a virtual machine).

Table 5. Compute Attributes

Attribute	Type	Description
OCCI-Compute-CPU-Arch	Enum (x86, x64)	CPU Architecture (e.g. x64)
OCCI-Compute-CPU-Cores	Integer	Number of CPU cores (e.g. 1, 2)
OCCI-Compute-CPU-Speed	Float (10^9 Hertz)	Clock speed in gigahertz (e.g. 2.4)
OCCI-Compute-Memory-Size	Float (10^6 bytes)	RAM in megabytes (e.g. 8192)
OCCI-Compute-Memory-Speed	Float (10^9 bytes/second)	RAM speed in Gbit/s (e.g. 17 for PC-8500 DDR3 per Wikipedia)
OCCI-Compute-Memory-Reliability	Enum (standard, checksum)	Qualitative measure of RAM reliability (e.g. ECC)

5.1.2. Network

A network resource is capable of transferring data (e.g. a virtual network or VLAN).

Table 6. Network Attributes

Attribute	Type	Description
`OCCI-Network-VLAN`	Integer (0..4095)	802.1q VLAN ID (e.g. `4095`)
`OCCI-Network-Label`	Token	Tag based VLANs (e.g. `external-dmz`)
`OCCI-Network-Address`	IPv4 or IPv6 Address (in CIDR notation)	IP gateway address or network address where there is none (e.g. `192.168.0.1/24`, `2001:db8:a::123/64`)
`OCCI-Network-Allocation`	Enum (`auto`, `dhcp`, `manual`)	Address allocation mechanism: • `auto` is handled automatically by infrastructure and/or guest agent • `dhcp` uses network-based allocation protocol(s) • `manual` requires preconfiguration or manual allocation

TODO: Tidy up network interface addressing.

5.1.3. Storage

A storage resource is capable of mass storage of data (e.g. a virtual hard drive).

Table 7. Storage Attributes

Attribute	Type	Description
`OCCI-Storage-Reliability`	Enum (transient, persistent, reliable)	Qualitative device persistence (e.g. transient)
`OCCI-Storage-Size`	Integer (10^9 bytes)	Drive size in gigabytes (e.g. `40`, `0.00144`)
`OCCI-Storage-Speed`	Integer (10^6 bytes/second)	Drive speed in MB/s (e.g. `600` for SAS/SATA-600 Wikipedia)

5.2. Extensions

Various extensions provide for more advanced management functionality such as billing, monitoring and reporting.

Bibliography

Normative References

Informative References

[Wikipedia] *Wikipedia: List of device bandwidths*. `http://en.wikipedia.org/wiki/List_of_device_bandwidths`. .

6. OCCI Registries

Table 8. HTTP Status Codes

Code	Description	Example
200 OK	Request completed successfully	Response is returned
201 Created	Request completed successfully, resource was created	Pointer to new resource returned
202 Accepted	Request accepted, processing not completed	Workload starting but not yet active
301 Moved Permanently	Resource has been assigned a new permanent URI	Workload migrated to another installation
302 Found	Resource resides temporarily under a different URI	Alias pointing to UUID can be updated
304 Not Modified	Conditional GET on resource that is unchanged	Client already has the latest version of the resource
400 Bad Request	Request could not be understood by the server due to malformed syntax	Client sent a representation that was unable to be understood
401 Unauthorized	The request requires user authentication	Client must retry with authentication
402 Payment Required	The server has refused to fulfill the request	Credit limit exceeded
403 Forbidden	The server understood the request, but is refusing to fulfill it	Attempt to access resource without permission
404 Not Found	The server has not found the resource	Feed or entry unknown
405 Method Not Allowed	The method specified is not allowed for the resource	Attempt to delete an immutable resource
406 Not Acceptable	The resource is not capable of requested content characteristics	Unsupported output format requested
409 Conflict	Request is in conflict with the current state of the resource	Resource updated by a third-party in the interim
410 Gone	Resource is gone, no forwarding address	Resource was deleted
500 Internal Server Error	Server encountered an unexpected condition	An unknown failure has occurred (e.g. out of memory)
501 Not Implemented	Functionality required to fulfill request is not implemented	A missing extension was called
502 Bad Gateway	An invalid response was received from an upstream server	The gateway received a malformed response from a node
503 Service Unavailable	Server is temporarily unable to handle the request	Server may be overloaded or down for maintenance
504 Gateway Timeout	No response was received from an upstream server	The gateway did not receive a response within the timeout period

TBD ...Clinton DeWittRobert
SyreMark NottinghamIan
HicksonDavid HyattLeonard
RichardsonSam RubyMark
NottinghamSam Johnston, TBD
Google, Inc. Apple, Inc.
September 16, 2009

7. Contributors

TBD...

8. OCCI Legal Notices

8.1. 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.

8.2. Disclaimer

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.

8.3. Full Copyright Notice