Open Cloud Computing Interface Specification

Status of This Document

This document provides information to the Cloud and Grid community. It is the second deliverable of the Open Cloud Computing Interface (OCCI) working group. Distribution is unlimited.

Copyright Notice

Abstract

The document describes a slim and extensible Interface for Infrastructre as a Service (IaaS) model based Clouds. The document consist of several modular parts which each can be used without the others. After a walkthrough the OCCI Core is described followed up by some renderings. Those parts can be seen as mandatory. For usage in IaaS based Clouds the Infrastructure part of OCCI is described next. Finally the Registries are summed up.

The OCCI Specification is very modulare and consist of several parts. OCCI core specification forms the foundation of this and upcoming specification parts. The focus for this deliverable is on virtual workloads which can be deployed on 'Infrastructure as a Service' based Clouds. Therefore the second part describes the OCCI Infrastructure. Also a rendering is provided which makes this implementation ready to implement.

For future developements renderings might change. As well as other parts of the Cloud stack can be described using the OCCI Core as a foundation.

Introduction

An overview of the document.

OCCI Core Specification

Introduction

The Open Cloud Computing Interface (OCCI) is an open protocol for all cloud computing services. A RESTful interface, it deviates from the underlying HyperText Transfer Protocol (HTTP) only where absolutely necessary and can be described as a "Resource Oriented Architecture (ROA)".RWS Unlike other envelope-based protocols which operate in-band, all existing HTTP features are available for caching, proxying, gatewaying and other advanced functionality such as partial GETs.

Each resource is identified by URL(s) and has one or more native representations as well as an XHTML5 rendering for direct end-user accessibility with embedded semantic web markup. As such OCCI can present both a machine interface (using native resource renderings) and a user interface (using HTML markup with forms and other web technologies such as Javascript/Ajax). HTTP content negotiation is used to select between alternative representations and metadata including associations between resources is exposed via HTTP headers (e.g. the `Link:` and `Category:` headers).

In this way OCCI is not responsible for the representations themselves, rather it enables users to organise and group resources together to build arbitrarily complex systems of inter-related resources. It relies on existing standards for rendering and does not make any recommendations of one standard format over any other.

Tip

This is the case for the World Wide Web today where many image, video and other supporting formats co-exist. Browsers support a number of the common formats and users choose the most appropriate for the task.

Example

```
> GET /us-east/webapp/vm01 HTTP/1.1
> User-Agent: occi-client/1.0 (linux) libcurl/7.19.4 OCCI/1.0
> Host: cloud.example.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sat, 10 Oct 2009 12:56:51 GMT
< Content-Type: application/ovf
< Link: </us-east/webapp/vm01;start>;
<       rel="http://purl.org/occi/action#start";
<       title="Start"
< Link: </us-east/webapp/build.pdf>;
<       rel="related";
<       title="Documentation";
<       type="application/pdf"
< Category: compute;
<       label="Compute Resource";
<       scheme="http://purl.org/occi/kind"
< Server: occi-server/1.0 (linux) OCCI/1.0
< Connection: close
<
< <?xml version="1.0" encoding="UTF-8"?>
...
```

Basics

Entry point

The interface is defined by a single URL entry point which will either be a *collection*, contain *link*(s) to *collection*(s) or both. All resources *must* be discoverable via [chains of] links from the entry point.

Kinds, Actions & 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 the REST hypertext constraint (whereby state transitions are defined *in-band* rather than via rules).

HTTP Verbs

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 MK-COL, MOVE and COPY.

POST (Create)	“The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordi-

	nate 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
MKCOL (Make Collection)	“MKCOL creates a new collection resource at the location specified by the Request-URI.”RFC4918
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

Connection

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 (such as server-side state) in line with RESTful principles.RFC2109

Versioning

Servers and clients *should* indicate the latest version of OCCI they support (e.g. `1.0`) by way of the `Server:` and `User-Agent:` headers respectively, using the token “`OCCI`” (e.g. “`OCCI/1.0`”). If none is provided the latest available version *shall* be used.

Model

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

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.

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 dot character (“.”).

Tip

This scalable approach was derived from the Mozilla Firefox `about:config` page.

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 based on their reversed Internet domain (e.g. “`com.cisco.cdp`”).

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.

Registry Entries

Table 1. Core Attributes

Attribute	Description	Type	Example
`id`	Immutable, unique identifier for the resource	URI	`urn:uuid:d0e9f0d0-f62d-4f28-bc90-23b0bd871770`
`title`	Display name for the resource	String	`Compute Resource #123`
`summary`	Description of the resource	String	`A virtual compute resource`
`version`	Specification version	Float	`1.0`

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: </us-east/webapp/vm01;start>;
      rel="http://purl.org/occi/action/start";
      title="Start"
```

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`). In the case of HTML-based renderings the actions can therefore be actual HTML forms.

Tip

Some resources can be interacted with but not rendered due to the nature of the resource or prevailing security policies (for example, an operator may be able to backup a machine without knowing anything about it).

Asynchronous Actions

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.

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.

Advanced Actions

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`.

Meta-model

The meta-model defines how objects interrelate.

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 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.

Examples

```
Category: compute;
      label="Compute Resource";
      scheme="http://purl.org/occi/kind/"
```

Querying

TODO: Pull query interface from GData: http://code.google.com/apis/gdata/docs/2.0/reference.html#Queries

Registry Entries

Table 2. Core Category Schemes

Scheme	Description
`http://purl.org/occi/kind/`	OCCI Kinds

Collections

Where an operation could return multiple resources (e.g. categories, searches) this is referred to as a *collection*. Collections are returned as a list of URLs in `text/uri-list` format.RFC2483

Tip

Collections are passed by reference for simplicity rather than performance reasons, requiring O(n+1) requests. Including metadata (via a wrapper format like Atom or SOAP) and/or the data itself would provide O(1) performance, though this "pass by value" approach should only be considered where the representations are known to be small as encoding adds significant overhead.

Examples

```
# OCCI Example Collection
/examples/custom-extension
/examples/lamp-multi-vm
/examples/lamp
/examples/myservice
```

Advertising

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

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

Tip

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

Paging

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

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

Linking

Web linking standards for 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: </us-east/webapp/build.pdf>;
      rel="related";
      title="Documentation";
      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.

Registry Entries

Table 3. Core Link Relations

Relation	Description
`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]

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).

Foreign markup

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

Security Considerations

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

Glossary

in-band	"Sending of metadata and control information in the same band, on the same channel, as used for data", for example, by em-

	bedding 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.

Bibliography

Normative References

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

[RFC2483] *RFC 2483 - URI Resolution Services Necessary for URN Resolution*. `http://tools.ietf.org/html/rfc2483#section-5 [http://tools.ietf.org/html/rfc2109]`. Internet Engineering Task Force (IETF) 1999-01.

[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 [http://tools.ietf.org/html/rfc2616]`. 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 [http://tools.ietf.org/html/rfc2616]`. Internet Engineering Task Force (IETF) 2007-06.

[OpenSearch] *OpenSearch 1.1*. `http://www.opensearch.org/Specifications/OpenSearch/1.1 [http://tools.ietf.org/html/rfc2616]`. 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.

[HTML5] *HTML 5*. `http://www.w3.org/TR/html5/ [http://tools.ietf.org/html/rfc4287]`. Ian Hickson. David Hyatt. World Wide Web Consortium (W3C) 2009-08-25.

[OAuth] *OAuth*. `http://oauth.net/core/1.0 [http://tools.ietf.org/html/rfc2616]`. 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.

OCCI Infrastructure

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

Table 4. Common Attributes

Attribute	**Type**	**Description**	**Mandatory/Optional**
`OCCI-Infrastructure-Hostname`	String	Valid DNS hostname for the resource (may be FQDN)	Mandatory

Kinds

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

Table 5. Kinds

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

Compute

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

Table 6. Compute Attributes

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

Network

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

Table 7. Network Attributes

Attribute	Type	Description	Mandatory/Optional
`OCCI-Network-VLAN`	Integer (0..4095)	802.1q VLAN ID (e.g. `4095`)	Optional
`OCCI-Network-Label`	Token	Tag based VLANs (e.g. `external-dmz`)	Optional
`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`)	Mandatory
`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	Mandatory

TODO: Tidy up network interface addressing.

Storage

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

Table 8. Storage Attributes

Attribute	Type	Description	Mandatory/Optional
`storage.retention-duration`	Time	The duration the storage instance is retained after deletion	Optional
`storage.capacity`	Integer plus "IEC60027-2 unit prefix"	Maximum size of the storage resource.	Mandatory
`storage.size`	Integer plus "IEC60027-2 unit prefix"	Size of the storage resource capacity in use.	Mandatory
`storage.status`	Enum (Online, Offline, Standby, Degraded)	The current operating status of the storage resource	Optional
`storage.class`	Enum (Volatile, Maintained)	The operating class of storage	Mandatory

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`. .

OCCI Machine Interface (HTTP)

2009-10-07

Specification

The HTTP binding for OCCI provides a machine interface, delivering resources in their native formats:

- The HTTP binding is defined by RFC2616 (HTTP).
- Web Linking [LINK] and Web Categories [CATEGORY] specifications are used for the meta-model.
- Server-side cookies ("Attributes") are used for name-value pairs.
- Collections are transferred as the `text/uri-list` content type.RFC2483

In all cases the same process as for Cookies (defined in RFC2965) is used to set/get headers, where `[Set-]Attribute:`, `[Set-]Category:` and `[Set-]Link:` are used in place of `Cookie:` and `Set-Cookie:`. `Set-*` headers may also be included on PUT or POST requests (including empty POSTs in order to update the metadata independently of the representation).

Existing values can be discarded by sending a `Set-*` header with the `discard` attribute and updated atomically by providing a new value in the same HTTP transaction.

Example

POST Request

```
POST /compute/123 HTTP/1.1
Host: example.com
Content-Length: 0
Set-Attribute: id="urn:uuid:d0e9f0d0-f62d-4f28-bc90-23b0bd871770"
Set-Category: compute;
  scheme="http://purl.org/occi/kind/";
  label="Compute Resource"
Set-Link: <http://example.com/products/1234>;
  rel="alternate";
  title="Alternate representation"
```

GET Response

```
Attribute: id="urn:uuid:d0e9f0d0-f62d-4f28-bc90-23b0bd871770"
```

```
Attribute: title="Compute Resource #123"
Attribute: summary="A virtual compute resource"
Attribute: updated="2009-12-31T12:59:59Z"
Attribute: compute.cores=2
Attribute: compute.speed=3000
Attribute: compute.memory=2048
ETag: "dad86c61eea237932f"
Category: compute;
  scheme="http://purl.org/occi/kind/";
  label="Compute Resource"
Link: <http://example.com/products/1234>;
  rel="alternate";
  title="Alternate representation"

<?xml version="1.0" encoding="UTF-8"?>
<ovf:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ovf="http://schemas.dmtf.org/ovf/1/envelope"
<!-- snip -->
```

Bibliography

Normative References

[RFC2483] *RFC 2483 - URI Resolution Services Necessary for URN Resolution.* `http://tools.ietf.org/html/rfc2483#section-5` [`http://tools.ietf.org/html/rfc2109`]. Internet Engineering Task Force (IETF) 1999-01.

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

[RFC2965] *RFC 2965 - HTTP State Management Mechanism.* `http://tools.ietf.org/html/rfc2965` [`http://tools.ietf.org/html/rfc2822`]. Internet Engineering Task Force (IETF) 2000-10.

Informative References

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

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

[HTML5-article] *Designing a great HTTP API - why heavyweight XML is not the answer.* `http://www.elastichosts.com/blog/2009/01/01/designing-a-great-http-api/` [`http://www.smashingmagazine.com/2009/07/29/misunderstanding-markup-xhtml-2-comic-strip/`]. . 2009-01-01.

Contributors

Next to the members of the OCCI working group the following people actively contributed to this document:

Table 9. List of contributors

Name	Affiliation	Contact
Andy Edmonds	Intel - SLA@SOI project	andrewx.edmonds@intel.com
Sam Johnston (Editor)	Australian Online Solutions	samj@samj.net

Name	Affiliation	Contact
Gary Mazzaferro	OCCI Counselour - Exxia, Inc.	garymazzaferro@gmail.com
Thijs Metsch (Editor)	Sun Microsystems - RESERVOIR project	thijs.metsch@sun.com
Andre Merzky	Lousiana State University	andre@merzky.net

OCCI Legal Notices

Intellectual Property Statement

Disclaimer

Full Copyright Notice