Open Cloud Computing Interface – Text Rendering

Status of this Document
This document provides information to the community regarding the specification of the Open Cloud Computing Interface. Distribution is unlimited.
This document obsoletes GFD-P-R.185.

Copyright Notice

Trademarks
OCCI is a trademark of the Open Grid Forum.

Abstract
This document, part of a document series produced by the OCCI working group within the Open Grid Forum (OGF), provides a high-level definition of a Protocol and API. The document is based upon previously gathered requirements and focuses on the scope of important capabilities required to support modern service offerings.
Contents

1 Introduction 4

2 Notational Conventions 4

3 Text rendering 5

4 ABNF Definitions 5
   4.1 Category ABNF 5
   4.2 Link ABNF 5
   4.3 Attribute ABNF 6
   4.4 Location ABNF 6

5 Renderings 6
   5.1 Entity Instance Rendering 6
      5.1.1 Resource Instance Rendering 6
      5.1.2 Link Instance Rendering 7
   5.2 Category Instance Rendering 7
      5.2.1 Kind Instance Rendering 7
      5.2.2 Mixin Instance Rendering 7
      5.2.3 Action Instance Rendering 7
   5.3 Entity Collection Rendering 7
      5.3.1 Resource Collection Rendering 7
      5.3.2 Link Collection Rendering 7
   5.4 Category Collection Rendering 8
      5.4.1 Kind Collection Rendering 8
      5.4.2 Mixin Collection Rendering 8
      5.4.3 Action Collection Rendering 8
   5.5 Attributes Rendering 8
      5.5.1 Entity Instance Attribute Rendering Specifics 8
      5.5.2 Mixin Instance Attribute Rendering Specifics 8
      5.5.3 Attribute Description Rendering 8

6 OCCI Text Plain rendering 9
   6.1 Example 9

7 OCCI Header Rendering 9
   7.1 Example 10

8 URI Listing Rendering 10

9 Security Considerations 10

10 Glossary 11

occi-wg@ogf.org 2
1 Introduction

The Open Cloud Computing Interface (OCCI) is a RESTful Protocol and API for all kinds of management tasks. OCCI was originally initiated to create a remote management API for IaaS model-based services, allowing for the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. It has since evolved into a flexible API with a strong focus on interoperability while still offering a high degree of extensibility. The current release of the Open Cloud Computing Interface is suitable to serve many other models in addition to IaaS, including PaaS and SaaS.

In order to be modular and extensible the current OCCI specification is released as a suite of complementary documents, which together form the complete specification. The documents are divided into four categories consisting of the OCCI Core, the OCCI Protocols, the OCCI Renderings and the OCCI Extensions.

- The OCCI Core specification consists of a single document defining the OCCI Core Model. OCCI interaction occurs through renderings (including associated behaviors) and is expandable through extensions.
- The OCCI Protocol specifications consist of multiple documents, each describing how the model can be interacted with over a particular protocol (e.g. HTTP, AMQP, etc.). Multiple protocols can interact with the same instance of the OCCI Core Model.
- The OCCI Rendering specifications consist of multiple documents, each describing a particular rendering of the OCCI Core Model. Multiple renderings can interact with the same instance of the OCCI Core Model and will automatically support any additions to the model which follow the extension rules defined in OCCI Core.
- The OCCI Extension specifications consist of multiple documents, each describing a particular extension of the OCCI Core Model. The extension documents describe additions to the OCCI Core Model defined within the OCCI specification suite.

The current specification consists of seven documents. This specification describes version 1.2 of OCCI and is backward compatible with 1.1. Future releases of OCCI may include additional protocol, rendering and extension specifications. The specifications to be implemented (MUST, SHOULD, MAY) are detailed in the table below.

<table>
<thead>
<tr>
<th>Document</th>
<th>OCCI 1.1</th>
<th>OCCI 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Model</td>
<td>MUST</td>
<td>MUST</td>
</tr>
<tr>
<td>Infrastructure Model</td>
<td>SHOULD</td>
<td>SHOULD</td>
</tr>
<tr>
<td>Platform Model</td>
<td>MAY</td>
<td>MAY</td>
</tr>
<tr>
<td>SLA Model</td>
<td>MAY</td>
<td>MAY</td>
</tr>
<tr>
<td>HTTP Protocol</td>
<td>MUST</td>
<td>MUST</td>
</tr>
<tr>
<td>Text Rendering</td>
<td>MUST</td>
<td>MUST</td>
</tr>
<tr>
<td>JSON Rendering</td>
<td>MAY</td>
<td>MUST</td>
</tr>
</tbody>
</table>

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].

1Infrastructure as a Service

occi-wg@ogf.org
3 Text rendering

This document presents the text-based renderings. To be complaint, OCCI implementations MUST implement the three renderings defined in sections 6, 7, and 8.

The following specification of the text-based renderings is in the process of being deprecated and will be removed or significantly changed in the next MAJOR release of the standard.

The document is structured by defining base ABNFs, which can then be combined into renderings, which will be rendered over a protocol (e.g., HTTP) by the specific rendering definitions.

4 ABNF Definitions

For the following section of 5 these ABNF notations will be used. Implementations MUST hence implement the renderings according to these definitions.

4.1 Category ABNF

The following syntax MUST be used for Category renderings:

```
Category = "Category" ";" #category-value
  category-value = term
  | ;" scheme" "=" <" scheme ">
  | ;" class" "=" ( class | <" class ">
  | [ ;" title" "=" quoted-string ]
  | [ ;" rel" "=" <" type-identifier ">
  | [ ;" location" "=" <" URI ">
  | [ ;" attributes" "=" <" attribute-list ">
  | [ ;" actions" "=" <" action-list ">
  term = ( ALPHA | DIGIT ) *( ALPHA | DIGIT | "-" | ";" | ";" )
  scheme = URI
  type-identifier = scheme term
  class = "action" | "mixin" | "kind"
  attribute-list = attribute-def
  | attribute-def *( 1*SP attribute-def)
  attribute-def = attribute-name
  | attribute-name
  | "{" attribute-property *( 1*SP attribute-property ) "}"
  attribute-property = "immutable" | "required"
  attribute-name = term
  action-list = action
  | action *( 1*SP action )
  action = type-identifier
```

4.2 Link ABNF

The following syntax MUST be used to represent OCCI Link type instance references:

```
Link = "Link" ";" #link-value
  link-value = "<" URI-reference ">
  | ;" rel" "=" <" resource-type ">
  | [ ;" self" "=" <" link-instance ">
  | [ ;" category" "=" link-type
  | *( ;" link-attribute )
```
term = ( ALPHA | DIGIT ) *( ALPHA | DIGIT | "-" | "_" | "." )
scheme = URI
type-identifier = scheme term
resource-type = type-identifier *( 1*SP type-identifier )
link-type = type-identifier *( 1*SP type-identifier )
link-instance = URI-reference
link-attribute = attribute-name "=" ( token | quoted-string )
attribute-name = term

The following syntax MUST be used to represent OCCI Action instance references:

ActionLink = "Link" ":" #link-value
  link-value = "<" action-uri ">"
  term = ( ALPHA | DIGIT ) *( ALPHA | DIGIT | "-" | "_" | "." )
scheme = URI
type-identifier = scheme term
action-type = type-identifier
action-uri = URI "?" "action=" term

4.3 Attribute ABNF

Attribute = "X-OCCI-Attribute" ":" #attribute-repr
attribute-repr = attribute-name "=" attribute-value
attribute-name = ( ALPHA | DIGIT ) *( ALPHA | DIGIT | "-" | "_" | "." )
attribute-value = ( string | number | bool | enum-val )
  string = quoted-string
  number = ( int | float)
  int = *DIGIT
  float = *DIGIT "." *DIGIT
  bool = ("true" | "false")
  enum-val = string

4.4 Location ABNF

Location = "X-OCCI-Location" ":" location-value
  location-value = URI-reference

5 Renderings

The renderings defined in this section will be used in the specific text rendering defined in section 6 and 7

5.1 Entity Instance Rendering

Entity instances MUST be rendered according to the following definitions.

5.1.1 Resource Instance Rendering

A Resource instance MUST be rendered using the following definition:

    resource_rendering = 1*( Category CRLF )
                         *( Link CRLF )
                         *( Attribute CRLF )
The rendering of a Resource instance MUST represent any associated Action instances using the ActionLink CRLF.

5.1.1.1 Action Invocation Rendering

Upon an Action invocation the client MUST send along the following definition:

\[
\text{action_definition} = 1(\text{Category CRLF}) \\
* (\text{Attribute CRLF})
\]

5.1.2 Link Instance Rendering

A Link instance MUST be rendered using the following definition:

\[
\text{link_rendering} = 1* (\text{Category CRLF}) \\
* (\text{ActionLink CRLF}) \\
* (\text{Attribute CRLF})
\]

5.2 Category Instance Rendering

A Category instances MUST be rendered as defined below.

5.2.1 Kind Instance Rendering

A Kind instance MUST be rendered as a Category CRLF.

5.2.2 Mixin Instance Rendering

A Mixin instance MUST be rendered as a Category CRLF.

5.2.3 Action Instance Rendering

An Action instance MUST be rendered as a Category CRLF.

Note that an Action instance MUST NOT have Link and Actions references.

5.3 Entity Collection Rendering

A collection of Resource or Link instances MUST be rendered as following:

\[
\text{entity_collection_rendering} = * (\text{Location CRLF})
\]

5.3.1 Resource Collection Rendering

see above

5.3.2 Link Collection Rendering

see above
5.4 Category Collection Rendering

For the Query interface the following Category instance rendering MUST be used:

\[
\text{category\_collection\_rendering} = *\text{( Category CRLF )}
\]

5.4.1 Kind Collection Rendering

see above

5.4.2 Mixin Collection Rendering

see above

5.4.3 Action Collection Rendering

see above

5.5 Attributes Rendering

5.5.1 Entity Instance Attribute Rendering Specifics

For Entity instances the following model attribute name to attribute name rendering mappings MUST be used:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Attribute name once rendered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity.id</td>
<td>occi.core.id</td>
</tr>
<tr>
<td>Entity.title</td>
<td>occi.core.title</td>
</tr>
<tr>
<td>Resource.summary</td>
<td>occi.core.summary</td>
</tr>
<tr>
<td>Link.target</td>
<td>occi.core.target</td>
</tr>
<tr>
<td>Link.target.kind</td>
<td>occi.core.target.kind</td>
</tr>
<tr>
<td>Link.source</td>
<td>occi.core.source</td>
</tr>
<tr>
<td>Link.source.kind</td>
<td>occi.core.source.kind</td>
</tr>
</tbody>
</table>

5.5.2 Mixin Instance Attribute Rendering Specifics

When rendering Mixin.depends and Mixin.applies to the rel attribute in the Category instance rendering, only Mixin.depends value MUST be used. If Mixin.depends contains multiple values, only the first value MUST be used.

5.5.3 Attribute Description Rendering

Attributes MUST be rendered as defined by the Attribute CRLF. If used, the pattern model attribute MUST be represented as a string in the ERE \[2\] format.
6 OCCI Text Plain rendering

The OCCI Text plain rendering specifies a rendering of OCCI instance types in a simple text format. The rendering can be used to render OCCI instances independently of the protocol being used. Thus messages can be delivered by, e.g., the HTTP protocol as specified in [3].

The following media-types MUST be used for the OCCI Text plain rendering:

- text/occi+plain
- text/plain

Each entry in the body consists of a name followed by a colon (":") and the field value.

6.1 Example

The following example show an Entity instance rendering using the Text plain rendering.

```xml
< Category: compute; \\
<    scheme="http://schemas.ogf.org/occi/infrastructure#" \\
<    class="kind"; \\
<    Link: </users/foo/compute/b9ff813e-fee5-4a9d-b839-673f39746096?action=start>; \\
<    rel="http://schemas.ogf.org/occi/infrastructure/compute/action#start"
< X-OCCI-Attribute: occi.core.id="urn:uuid:b9ff813e-fee5-4a9d-b839-673f39746096"
< X-OCCI-Attribute: occi.core.title="My Dummy VM"
< X-OCCI-Attribute: occi.compute.architecture="x86"
< X-OCCI-Attribute: occi.compute.state="inactive"
< X-OCCI-Attribute: occi.compute.speed=1.33
< X-OCCI-Attribute: occi.compute.memory=2.0
< X-OCCI-Attribute: occi.compute.cores=2
< X-OCCI-Attribute: occi.compute.hostname="dummy"
```

7 OCCI Header Rendering

The following media-type MUST be used for the OCCI header Rendering:

- text/occi

While using this rendering the HTTP Protocol [3] MUST be used and the renderings MUST be placed in the HTTP Header. The body MUST contain the string "OK" on successful operations.

The HTTP header fields MUST follow the specification in RFC 7230 [4]. A header field consists of a name followed by a colon (":") and the field value.

**Limitations:** HTTP header fields MAY appear multiple times in a HTTP request or response. In order to be OCCI compliant, the specification of multiple message-header fields according to RFC 7230 MUST be fully supported. In essence there are two valid representations of multiple HTTP header field values. A header field might either appear several times or as a single header field with a comma-separated list of field values. Due to implementation issues in many web frameworks and client libraries it is RECOMMENDED to use the comma-separated list format for best interoperability.

HTTP header field values, which contain separator characters, MUST be properly quoted according to RFC 7230. Space in the HTTP header section of a HTTP request is a limited resource. By this, it is noted that many HTTP servers limit the number of bytes that can be placed in the HTTP header area. Implementers MUST be aware of this limitation in their own implementations and take appropriate measures so that truncation of header data does NOT occur.
7.1 Example

The following example shows an Entity instance rendering using the Text header rendering.

```xml
< Category: compute; 
    scheme="http://schemas.ogf.org/occi/infrastructure#" 
    class="kind";
< Link: </users/foo/compute/b9ff813e-fee5-4a9d-b839-673f39746096?action=start>; 
    rel="http://schemas.ogf.org/occi/infrastructure/compute/action#start"
< X-OCCI-Attribute: occi.core.id="urn:uuid:b9ff813e-fee5-4a9d-b839-673f39746096", 
    occi.core.title="My Dummy VM", occi.compute.architecture="x86", 
    occi.compute.state="inactive", occi.compute.speed=1.33, 
    occi.compute.memory=2.0, occi.compute.cores=2, 
    occi.compute.hostname="dummy"
< OK
```

8 URI Listing Rendering

The following media-types MUST be used for the URI Rendering:

text/uri-list

This rendering cannot render resource instances or Kinds or Mixins directly but just links to them. For concrete rendering of Kinds and Categories the Content-types text/occi, text/plain MUST be used. If a request is done with the text/uri-list in the Accept header, while not requesting for a Listing a Bad Request MUST be returned. Otherwise a list of resources MUST be rendered in text/uri-list format as defined in [5], which can be used for listing resource in collections or the name-space of the OCCI implementation.

9 Security Considerations

OCCI does not require that an authentication mechanism be used nor does it require that client to service communications are secured. It does RECOMMEND that an authentication mechanism be used and that where appropriate, communications are encrypted using HTTP over TLS. The authentication mechanisms that MAY be used with OCCI are those that can be used with HTTP and TLS. For further discussion see the appropriate section in [3].
# Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>An OCCI base type. Represents an invocable operation on an Entity sub-type instance or collection thereof.</td>
</tr>
<tr>
<td>Attribute</td>
<td>A type in the OCCI Core Model. Describes the name and properties of attributes found in Entity types.</td>
</tr>
<tr>
<td>Category</td>
<td>A type in the OCCI Core Model and the basis of the OCCI type identification mechanism. The parent type of Kind.</td>
</tr>
<tr>
<td>capabilities</td>
<td>In the context of Entity sub-types capabilities refer to the Attributes and Actions exposed by an entity instance.</td>
</tr>
<tr>
<td>Collection</td>
<td>A set of Entity sub-type instances all associated to a particular Kind or Mixin instance.</td>
</tr>
<tr>
<td>Entity</td>
<td>An OCCI base type. The parent type of Resource and Link.</td>
</tr>
<tr>
<td>entity instance</td>
<td>An instance of a sub-type of Entity but not an instance of the Entity type itself. The OCCI model defines two sub-types of Entity: the Resource type and the Link type. However, the term entity instance is defined to include any instance of a sub-type of Resource or Link as well.</td>
</tr>
<tr>
<td>Kind</td>
<td>A type in the OCCI Core Model. A core component of the OCCI classification system.</td>
</tr>
<tr>
<td>Link</td>
<td>An OCCI base type. A Link instance associates one Resource instance with another.</td>
</tr>
<tr>
<td>Mixin</td>
<td>A type in the OCCI Core Model. A core component of the OCCI classification system.</td>
</tr>
<tr>
<td>mix-in</td>
<td>An instance of the Mixin type associated with an entity instance. The “mix-in” concept as used by OCCI only applies to instances, never to Entity types.</td>
</tr>
<tr>
<td>OCCI</td>
<td>Open Cloud Computing Interface.</td>
</tr>
<tr>
<td>OGF</td>
<td>Open Grid Forum.</td>
</tr>
<tr>
<td>Resource</td>
<td>An OCCI base type. The parent type for all domain-specific Resource sub-types. See entity instance. This term is considered obsolete.</td>
</tr>
<tr>
<td>resource instance</td>
<td>A Mixin instance with no attributes or actions defined. Used for taxonomic organisation of entity instances.</td>
</tr>
<tr>
<td>tag</td>
<td>A Mixin instance which if associated at instance creation-time pre-populate certain attributes.</td>
</tr>
<tr>
<td>type</td>
<td>One of the types defined by the OCCI Core Model. The Core Model types are Category, Attribute, Kind, Mixin, Action, Entity, Resource and Link.</td>
</tr>
<tr>
<td>concrete type/sub-type</td>
<td>A concrete type/sub-type is a type that can be instantiated.</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier.</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator.</td>
</tr>
<tr>
<td>URN</td>
<td>Uniform Resource Name.</td>
</tr>
</tbody>
</table>

## Contributors

We would like to thank the following people who contributed to this document:

occi-wg@ogf.org
Next to these individual contributions we value the contributions from the OCCI working group.

12 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.

13 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.

14 Full Copyright Notice

Copyright © Open Grid Forum (2009-2016). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included as references to the derived portions on all such copies and derivative works. The published OGF document from which such works are derived, however, may not be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the purpose of developing new or updated OGF documents in conformance with the procedures defined in the OGF Document Process, or as required to translate it into languages other than English. OGF, with the approval of its board, may remove this restriction for inclusion of OGF document content for the purpose of producing standards in cooperation with other international standards bodies.
The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.

References


A Change Log

The corrections introduced by the September 19, 2016 update are summarized below. This section describes the possible impact of the corrections on existing implementations and associated dependent specifications.

- Relaxed rules on term values allowing the use of: alphanumerical characters (a-zA-Z0-9), “,” “-” and “.”.
- Explicitly stated how Mixin.depends and Mixin.applies should be rendered to rel on Mixin instances.