

1 Draft
2 OCCI-WG
3

Thijs Metsch, Intel
Mohamed Mohamed, Telecom SudParis
February 23, 2016

4 **Open Cloud Computing Interface – Platform**

5 Status of this Document

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

8 Copyright Notice

9 Copyright © Open Grid Forum (2014-2016). All Rights Reserved.

10 Trademarks

11 OCCI is a trademark of the Open Grid Forum.

12 Abstract

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

16	Contents	
17	1 Introduction	3
18	2 Notational Conventions	3
19	3 Platform	4
20	3.1 Application Kind Definition	4
21	3.2 Component Kind Definition	5
22	3.3 Linking to Components	6
23	3.4 Platform Templates	6
24	3.4.1 Application Template	6
25	3.4.2 Resource Template	7
26	4 Specific Component Instance Mixins	7
27	4.1 Database Mixin	7
28	4.1.1 Database Link	7
29	5 Security Considerations	8
30	6 Glossary	9
31	7 Contributors	9
32	8 Intellectual Property Statement	10
33	9 Disclaimer	10
34	10 Full Copyright Notice	10

1 Introduction

The Open Cloud Computing Interface (OCCI) is a RESTful Protocol and API for all kinds of management tasks. OCCI was originally initiated to create a remote management API for IaaS¹ model-based services, allowing for the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. It has since evolved into a flexible API with a strong focus on interoperability while still offering a high degree of extensibility. The current release of the Open Cloud Computing Interface is suitable to serve many other models in addition to IaaS, including PaaS and SaaS.

In order to be modular and extensible the current OCCI specification is released as a suite of 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 1. What OCCI specifications must be implemented for the specific version.

Document	OCCI 1.1	OCCI 1.2
Core Model	MUST	MUST
Infrastructure Model	SHOULD	SHOULD
Platform Model	MAY	MAY
SLA Model	MAY	MAY
HTTP Protocol	MUST	MUST
Text Rendering	MUST	MUST
JSON Rendering	MAY	MUST

OCCI makes an ideal interoperable boundary interface between the web and the internal resource management system of platform providers.

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

3 Platform

The OCCI Platform document details how an OCCI implementation can model and implement a Platform as a Service API offering by extending the OCCI Core Model. This API enables the provisioning and management of PaaS resources. For example, it allows to deploy an application on one or more PaaS components. The application itself could be composed of different components. The main platform types defined within OCCI Platform are:

Application Which defines the user-defined part of the overall service.

Component A configured instance of a piece of code providing business functions that are part of the execution of the application or responsible of hosting the application.

ComponentLink Connects an **Application** instance to a hosting **Component** or connects two components.

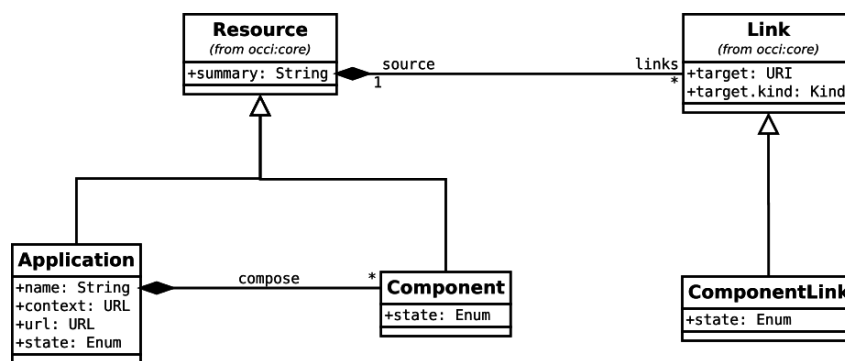


Figure 1. Overview Diagram of OCCI Platform Types.

These platform types inherit the OCCI Core Model Resource base type and all their attributes. One can use a suitable transport protocol (e.g., HTTP) and a suitable rendering to discover and consume these resources. Independently of the implementation, the defined resources could be discoverable during runtime through OCCI compliant interfaces.

As required by the OCCI Core Model specification, every instantiated type that is a sub-type of **Resource** or **Link** MUST be assigned a **Kind** that identifies the instantiated type. Each such **Kind** instance MUST be related to the Resource or Link base type's **Kind**. That assigned **Kind** instance MUST always remain immutable to any client.

3.1 Application Kind Definition

The following kind MUST be present and represents the kind definition of an application resource.

Application inherits the **Resource** base type defined in OCCI Core Model [2]. **Application** is assigned the **Kind** instance `http://schemas.ogf.org/occi/platform#application`. An **Application** instance MUST use and expose this **Kind**. The **Kind** instance assigned to the **Application** type MUST be related to the `http://schemas.ogf.org/occi/core#resource` **Kind** by setting the parent attribute.

Table 2 describes the **Attributes** defined by an **Application** instance. These attributes MAY or MUST be exposed by an instance of the **Application** type depending on the "Multiplicity" column in the aforementioned table.

The **Actions** are defined by the **Kind** instance `http://schemas.ogf.org/occi/platform#application`. Every **Action** instance in the table uses the `http://schemas.ogf.org/occi/platform/application/action/#` categorisation scheme. "Action Term" below refers to `Action.term`.

The state model for the **Application** instance is defined in Fig. 2.

Table 2. Attributes defined for the **Application** type.

Attribute	Type	Multiplicity	Mutability	Description
occi.app.name	String	1	Mutable	Name of the application.
occi.app.context	URL	1	Immutable	URL for contextualizing the app.
occi.app.url	URL	1	Immutable	DNS entry.
occi.app.state	Enum {active, inactive, error}	1	Immutable	State of the application.
occi.app.state.message	String	0..1	Immutable	Human-readable explanation of the current instance state.

Table 3. Actions applicable to instances of the **Application** type.

Action Term	Target state	Attributes
start	active	–
stop	inactive	–

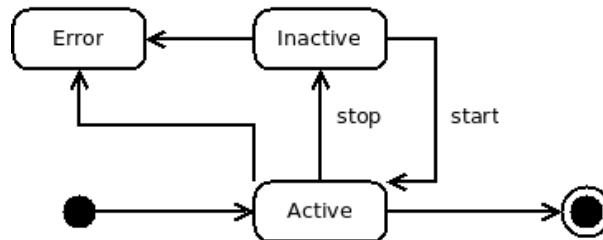


Figure 2. State model of an **Application** instance.

3.2 Component Kind Definition

The following kind MUST be present and represents the kind definition of a component resource.

Component inherits the **Resource** base type defined in OCCI Core Model [2]. **Component** is assigned the **Kind** instance <http://schemas.ogf.org/occi/platform#component>. A **Component** instance MUST use and expose this **Kind**. The **Kind** instance assigned to the **Component** type MUST be related to the <http://schemas.ogf.org/occi/core#resource> **Kind** by setting the parent attribute.

Table 4. Attributes defined for the **Component** type.

Attribute	Type	Multiplicity	Mutability	Description
occi.component.state	Enum {active, inactive, error}	1	Immutable	State of the component.
occi.component.state.message	String	0..1	Immutable	Human-readable explanation of the current instance state.

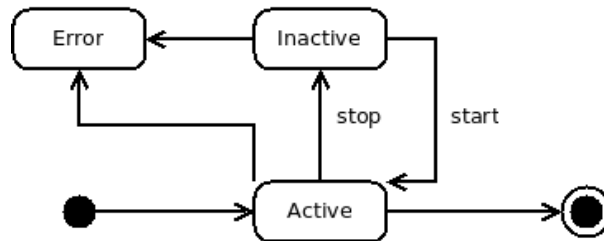
Table 4 describes the **Attributes** defined by **Component** instance. These attributes MAY or MUST be exposed by an instance of the **Component** type depending on the “Multiplicity” column in the aforementioned table.

The **Actions** are defined by the **Kind** instance <http://schemas.ogf.org/occi/platform#component>. Every **Action** instance in the table uses the <http://schemas.ogf.org/occi/platform/component/action#> categorisation scheme. “Action Term” below refers to **Action.term**

The state model for the **Component** instance is defined in Fig. 3.

Table 5. Actions applicable to instances of the **Application** type.

Action Term	Target state	Attributes
start	active	–
stop	inactive	–

**Figure 3.** State model of a **Component** instance.

112 3.3 Linking to Components

113 The composition of a service is realized through the linkage of **Application** and **Component** instances with
 114 each other. **Application** can be linked to many **Component** instances. This allows **Application** and **Components**
 115 to form acyclic graphs. To illustrate this with an example, the **Application** is the frontend, perceived by the
 116 user, to a composition of **Components**. To have a composition of **Components** (e.g. microservices) those
 117 **Components** need to be related to one another (e.g. Application linking to its DB or the DB linking to a
 118 monitoring service).

119 **ComponentLink** inherits the **Link** base type defined in OCCI Core Model [2]. **ComponentLink** is assigned
 120 the **Kind** instance <http://schemas.ogf.org/occi/platform#componentlink>. The **Kind** instance assigned to the
 121 **ComponentLink** type MUST be related to the <http://schemas.ogf.org/occi/core#link> **Kind** by setting the
 122 parent attribute.

123 The **ComponentLink** kind can be further enhanced by the use of provider-specific Mixins. This can be used to
 124 expose details such as database access URIs for an application linked up with a database component.

125 3.4 Platform Templates

126 Platform Templates allow for clients of an OCCI implementation to quickly and conveniently apply predefined
 127 configurations to OCCI Platform defined types. They are implemented using Mixin instances. There are two
 128 supported platform template types in OCCI Platform.

129 3.4.1 Application Template

130 Application templates allow clients to define which underlying framework the application should use (e.g.,
 131 Programming language).

132 The Application Template is defined by a Mixin. A provider-specific defined Application Template Mixin MUST
 133 relate to the OCCI Application Template Mixin through the `depends` attribute in order to give absolute type
 134 information. The OCCI Application Template is defined by the http://schemas.ogf.org/occi/platform#app_tpl
 135 Mixin and MUST be supported should Application Templates be offered.

136 Provider-specific Application Templates are constructed using a “term” and “scheme” combination where the
 137 “term” is a provider-specific description of the framework (e.g., python, ruby, ...). Where an implementation
 138 requires additional information to be held in the Templates Mixin, it MAY do so by using **Category**’s inherited
 139 **Attributes**.

140 **3.4.2 Resource Template**

141 The Resource Template Mixin builds upon the concept of Application Templates. A Resource Template is a
 142 provider defined Mixin instance that refers to a preset Resource configuration.

143 This can be used to define the resource instance attributes of the application and component. The provider-
 144 specific Resource Templates are defined by using a “term” and “scheme” combination. Those provider-specific
 145 Resource Template Mixin must relate to the OCCI Resource Template defined by *http://schemas.ogf.org/occi/
 146 platform#res_tpl* through the depends attribute. Where an implementation requires additional information to
 147 be held in the Templates Mixin, it MAY do so by using *Category*’s inherited *Attributes*.

148 An example of these templates is shown in the following UML diagram in Figure 4.

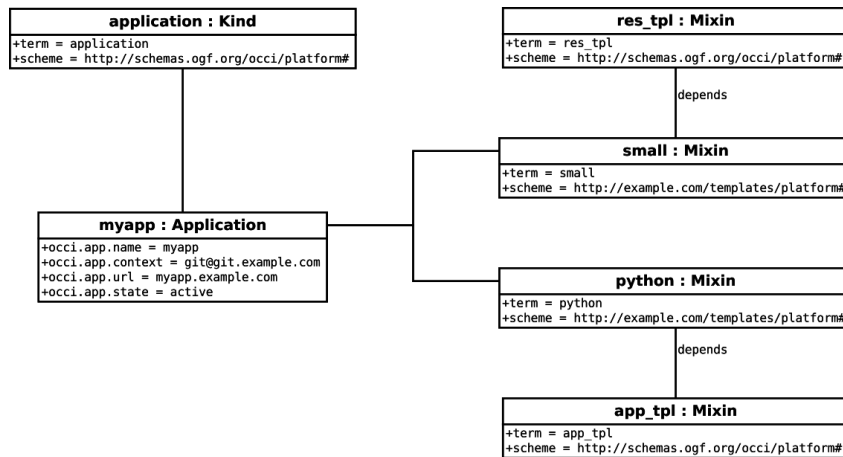


Figure 4. Application and Resource Templates.

149 **4 Specific Component Instance Mixins**

150 The following sections describe *Mixin* instances, which SHOULD be implemented by Providers for some basic
 151 component type.

152 **4.1 Database Mixin**

153 *Database* inherits the *Mixin* base type defined in OCCI Core Model [2]. *Database* is assigned the *Mixin* instance
 154 *http://schemas.ogf.org/occi/platform#database*. The *Database* instance applies to the *Component* instance
 155 defined above.

Table 6. Attributes defined for the *Database* type.

Attribute	Type	Multi- plicity	Mutability	Description
occi.database.version	String	1	Immutable	Version of the database.

156 Table 6 describes the *Attributes* defined by *Database* instance.

157 **4.1.1 Database Link**

158 In case that an *Application* instance links to a *Component* instance, which has the *Database Mixin* instance
 159 applied the following *Mixin* SHOULD be applied to the *ComponentLink*.

160 **DatabaseLink** inherits the **Mixin** base type defined in OCCI Core Model [2]. **DatabaseLink** is assigned the **Mixin**
161 instance `http://schemas.ogf.org/occi/platform#databaselink`. The **DatabaseLink** instance applies to the
162 **ComponentLink** instance defined above.

Table 7. **Attributes** defined for the **Database** type.

Attribute	Type	Multi- plicity	Mutability	Description
occi.database.uri	URI	1	Immutable	Connection URI for the database instance.
occi.database.username	URI	0 .. 1	Immutable	Username.
occi.database.token	URI	0 .. 1	Immutable	Token.

163 Table 7 describes the **Attributes** defined by **DatabaseLink** instance.

164 5 Security Considerations

165 The OCCI Platform specification is an extension to the OCCI Core Model specification [2]; thus the same
166 security considerations as for the OCCI Core Model specification apply here.

167 6 Glossary

Term	Description
Action	An OCCI base type. Represents an invocable operation on an Entity sub-type instance or collection thereof.
Attribute	A type in the OCCI Core Model. Describes the name and properties of attributes found in Entity types.
Category	A type in the OCCI Core Model and the basis of the OCCI type identification mechanism. The parent type of Kind .
capabilities	In the context of Entity sub-types capabilities refer to the Attributes and Actions exposed by an entity instance .
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 .
entity instance	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 <i>entity instance</i> is defined to include any instance of a sub-type of Resource or Link as well.
Kind	A type in the OCCI Core Model. A core component of the OCCI classification system.
168 Link	An OCCI base type. A Link instance associates one Resource instance with another.
Mixin	A type in the OCCI Core Model. A core component of the OCCI classification system.
mix-in	An instance of the Mixin type associated with an <i>entity instance</i> . The “mix-in” concept as used by OCCI <i>only</i> applies to instances, never to Entity types.
OCCI	Open Cloud Computing Interface.
OGF	Open Grid Forum.
Resource	An OCCI base type. The parent type for all domain-specific Resource sub-types.
resource instance	See <i>entity instance</i> . This term is considered obsolete.
tag	A Mixin instance with no attributes or actions defined. Used for taxonomic organisation of entity instances.
template	A Mixin instance which if associated at instance creation-time pre-populate certain attributes.
type	One of the types defined by the OCCI Core Model. The Core Model types are Category , Attribute , 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.
169 URN	Uniform Resource Name.

170 7 Contributors

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

Name	Affiliation	Contact
Andy Edmonds	ICCLab, ZHAW	edmo at zhaw.ch
Peter Troeger	TU Chemnitz	peter@troeger.eu
172 Thijs Metsch	Intel	thijs.metsch@intel.com
Sami Yangui	Concordia University	s_yangui@encs.concordia.ca
Mohamed Mohamed	Telecom SudParis	
Philippe Merle	Inria	philippe.merle@inria.fr

173 Next to these individual contributions we value the contributions from the OCCI working group.

174 **8 Intellectual Property Statement**

175 The OGF takes no position regarding the validity or scope of any intellectual property or other rights that
176 might be claimed to pertain to the implementation or use of the technology described in this document or the
177 extent to which any license under such rights might or might not be available; neither does it represent that
178 it has made any effort to identify any such rights. Copies of claims of rights made available for publication
179 and any assurances of licenses to be made available, or the result of an attempt made to obtain a general
180 license or permission for the use of such proprietary rights by implementers or users of this specification can be
181 obtained from the OGF Secretariat.

182 The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications,
183 or other proprietary rights which may cover technology that may be required to practice this recommendation.
184 Please address the information to the OGF Executive Director.

185 **9 Disclaimer**

186 This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all
187 warranties, express or implied, including but not limited to any warranty that the use of the information herein
188 will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

189 **10 Full Copyright Notice**

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

191 This document and translations of it may be copied and furnished to others, and derivative works that comment
192 on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in
193 whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph
194 are included on all such copies and derivative works. However, this document itself may not be modified in
195 any way, such as by removing the copyright notice or references to the OGF or other organizations, except
196 as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights
197 defined in the OGF Document process must be followed, or as required to translate it into languages other
198 than English.

199 The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or
200 assignees.

201 **References**

- 202 [1] S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels,” RFC 2119 (Best Current Practice),
203 Internet Engineering Task Force, Mar. 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2119.txt>
- 204 [2] R. Nyrén, A. Edmonds, A. Papaspyrou, and T. Metsch, “Open Cloud Computing Interface – Core,” Draft,
205 November 2015. [Online]. Available: TBD