GFD-R-P.227
OCCI-WG

Thijs Metsch, Intel
Mohamed Mohamed, Telecom SudParis
April 7, 2017

Open Cloud Computing Interface – Platform

Status of this Document

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

Copyright Notice

Trademarks

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

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[1] 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] Infrastructure 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 An 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.

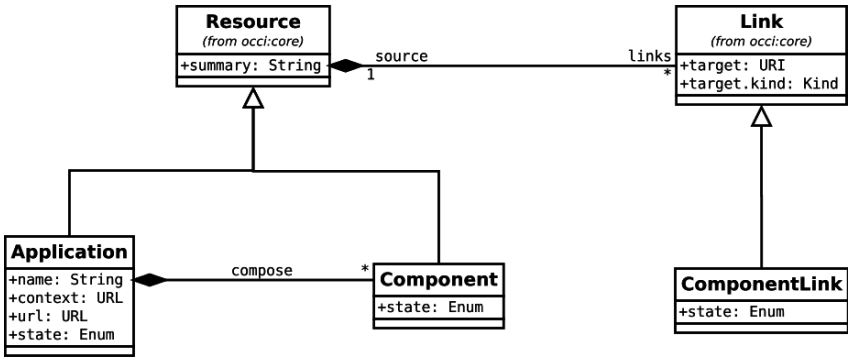

Figure 1. Overview Diagram of OCCI Platform Types.

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 [**?**]. 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	String	0..1	Immutable	Any data suitable for contextualizing the app.
occi.app.url	URL	0..1	Immutable	DNS entry.
occi.app.state	Enum {undeployed, deployed, 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
deploy	deployed	–
configure	inactive	–
start	active	–
stop	inactive	–
undeploy	undeployed	–

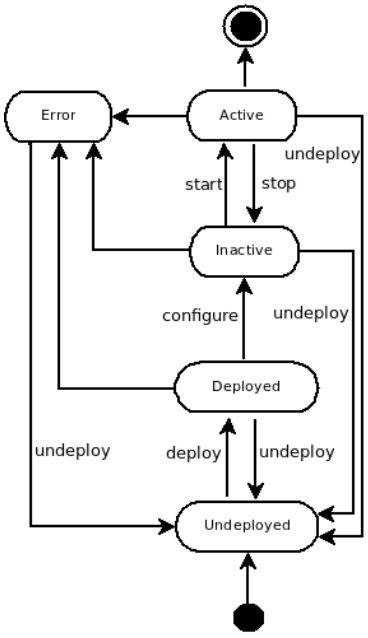

Figure 2. State model of an Application instance.

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 [**?**]. 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 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

Table 4. Attributes defined for the Component type.

Attribute	Type	Multiplicity	Mutability	Description
occi.component.context	String	0..1	Immutable	Any data suitable for contextualizing the component.
occi.component.state	Enum {undeployed, deployed, 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.

Action instance in the table uses the *http://schemas.ogf.org/occi/platform/component/action#* categorisation scheme. "Action Term" below refers to Action.`term`

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

Action Term	Target state	Attributes
deploy	deployed	–
configure	inactive	–
start	active	–
stop	inactive	–
undeploy	undeployed	–

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

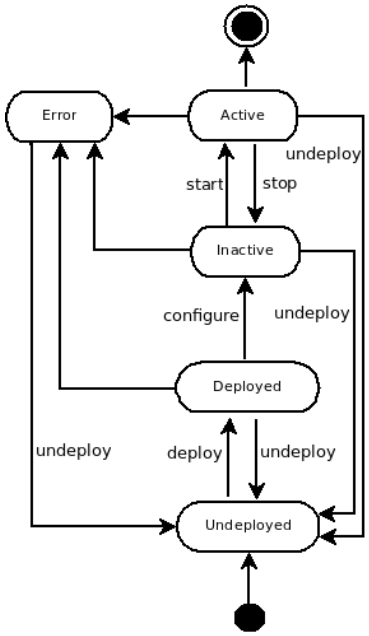

Figure 3. State model of a Component instance.

3.3 Linking to Components

The composition of a service is realized through the linkage of Application and Component instances with each other. Application can be linked to many Component instances. This allows Application and Components to form acyclic graphs. To illustrate this with an example, the Application is the frontend, perceived by the user, to a composition of Components. To have a composition of Components (e.g. microservices) those

Components need to be related to one another (e.g. Applciation linking to its DB or the DB linking to a monitoring service).

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

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

3.4 Platform Templates

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

3.4.1 Application Template

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

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

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

3.4.2 Resource Template

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

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

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

4 Specific Component Instance Mixins

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

4.1 Database Mixin

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

Table 6 describes the Attributes defined by Database instance.

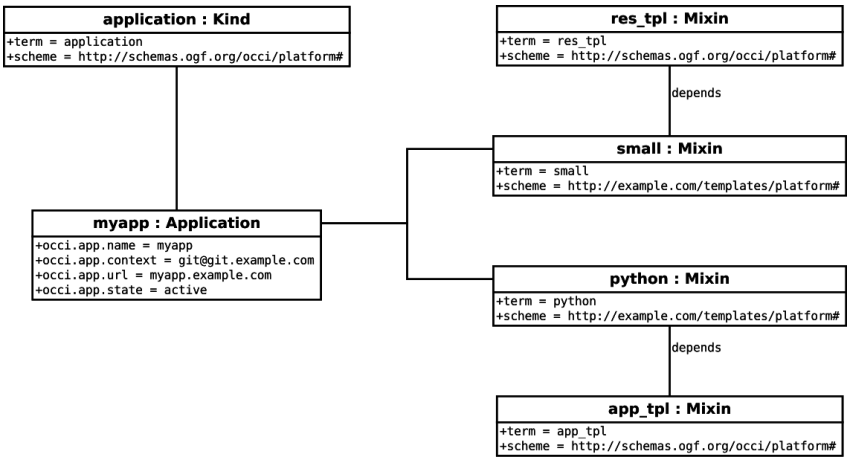

Figure 4. Application and Resource Templates.

Table 6. Attributes defined for the Database type.

Attribute	Type	Multiplicity	Mutability	Description
occi.database.version	String	1	Immutable	Version of the database.

4.1.1 Database Link

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

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

Table 7. Attributes defined for the Database type.

Attribute	Type	Multiplicity	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.

Table 7 describes the Attributes defined by DatabaseLink instance.

5 Security Considerations

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

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 *entity instance* 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.
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 *entity instance*. The "mix-in" concept as used by OCCI *only* 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 *entity instance*. 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.
URN	Uniform Resource Name.

7 Contributors

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
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
Pierre-Yves Gibello	Linagora	pygibello@linagora.com

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

8 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.

9 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.

10 Full Copyright Notice

11 Change Log

The corrections introduced by the April 7, 2017 update are summarized below.

- State diagrams for Application and Component (fig. 2 and 3) have new states and actions, to enhance their lifecycle (added "undeployed" / "deployed" states, and "deploy" / "undeploy" / "configure" actions).
- The semantic of Application's "context" attribute has been extended to any kind of context data, not only a URL (and the "context" field is now optional).
- A "context" attribute has been added to Component, with the same semantic as the "context" attribute of an Application.