Thijs Metsch (INTEL)
Mohamed Mohamed (Telecom SudParis)
Date 29.08.2014

Open Cloud Computing Interface Platform extension

Status of this Document

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

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

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 exible 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 three categories consisting of the OCCI Core, the OCCI Renderings and the OCCI Extensions.

- The OCCI Core specification consists of a single document defining the OCCI Core Model. The OCCI Core Model can be interacted with renderings (including associated behaviours) and expanded through extensions
- The OCCI Rendering specifications consist of multiple documents each describing 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. They do not require changes to the HTTP Rendering specifications as of this version of the specification.

This document describes an extension specification for a Platform as a Service (PaaS) model. If wanted it can be used together with the Infrastructure extension.

Notational Conventions

All these parts and the information within are mandatory for implementers (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].

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**: Software fragment resources.
- **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.

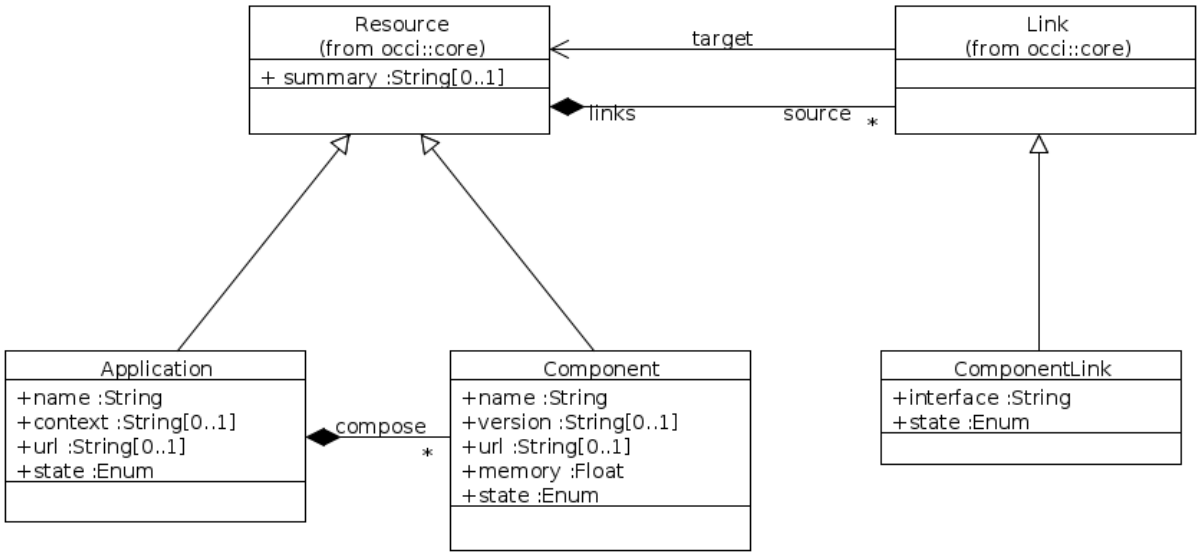

Figure 1. Overview Diagram of OCCI Platform Types

These platform types inherit the OCCI Core Model Resource base type and all their attributes. One can use the HTTP RESTful 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 type instantiated 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.

Application kind definitions

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

Scheme	*http://schemas.ogf.org/occi/platform#*
Term	*application*
Title	*A PaaS application*
rel	*http://schemas.ogf.org/occi/core#resource*
Attributes	
*occi.app.name (***required***)*	Name of the application
occi.app.context	URL for contextualizing the app
*occi.app.url (***immutable***)*	DNS entry
*occi.app.state (***immutable***)*	State of the application [‘active’, ‘inactive’, ‘updating’, ‘deployed’, ‘error’]
Actions	
start	Start the application
stop	Stop the application

The Figure 2 shows a state diagram of Application instances. It describes the different states and transitions of Application instance.

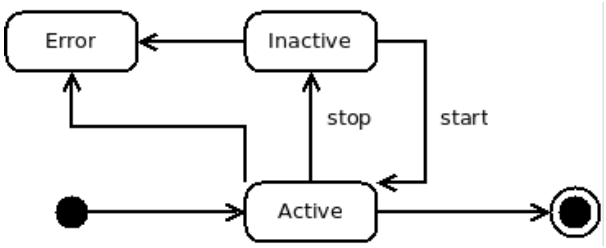

Figure 2. state model of the Application kind

The following kind defines components provided by the PaaS. This can be a component used by the application during its runtime (e.g. Databases such as MongoDB or MySQL).

Scheme	*http://schemas.ogf.org/occi/platform#*
Term	*component*
Title	*A component of a PaaS application*
rel	*http://schemas.ogf.org/occi/core#resource*

Attributes	
occi.component.state **(immutable)**	State of the component [‘active’, 'inactive', ’error’]
Actions	
start	Activate the component.
stop	Deactivate the component.

The states of the component are defined in the following state diagram in Figure 3:

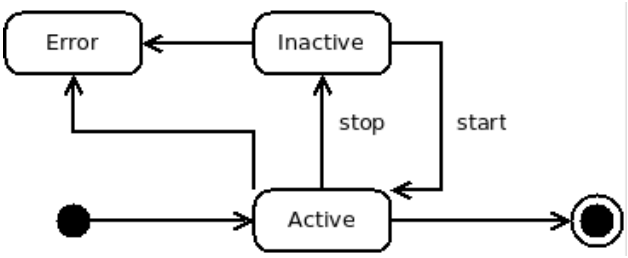

Figure 3. state model of the component kind.

Mixins can be applied to this kind which further define the capabilities of the component. Later these components can linked to other kinds as described by the mechanisms below.

Linking to components

The composition of a service is realized through the linkage of applications and components with each other. The following link definition MUST be implemented:

Scheme	*http://schemas.ogf.org/occi/platform#*
Term	*componentLink*
Title	*A link to a service component*
rel	*http://schemas.ogf.org/occi/core#link*
Attributes	
N/A	
Actions	
N/A	

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

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 templates types in OCCI Platform.

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 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 hold in the Templates Mixin, it MAY do so by using Category’s inherited Attributes.

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*. Where an implementation requires additional information to be hold 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.

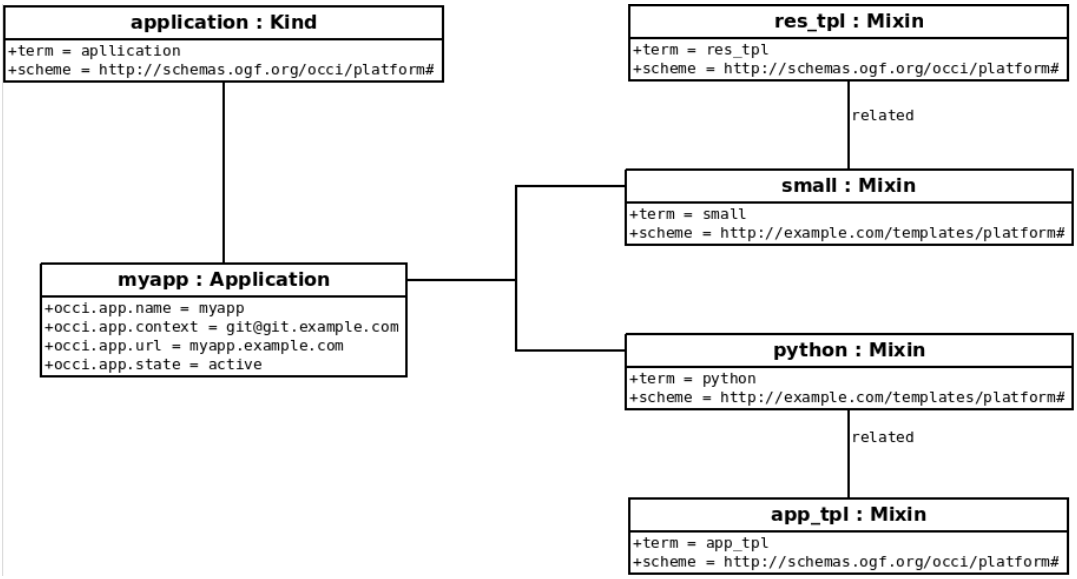

Figure 4. Object diagram of an Application instance and its associated Mixins and Kinds.

Security Considerations

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

Glossary

TODO

Contributors

- Andy Edmonds
- Peter Troeger
- Thijs Metsch
- Sami Yangui
- Mohamed Mohamed
- <ask for invite to doc to be added here>

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 sort 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.

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 completeness for a particular purpose.

Full Copyright Notice

References

[1] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels," RFC 2119 (Best Current Practice), Internet Engineering Task Force, Mar. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2119.txt
[2] OCCI Core.