GWD-R
OCCI-WG

Thijs Metsch, Platform Computing
Andy Edmonds, Intel
Ralf Nyrén, Aurenav
October 14, 2010
Updated: November 4, 2010

Open Cloud Computing Interface - Core

Status of this Document

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

Obsoletes

This document obsoletes GFD-xxx [REFERENCE].

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 model based Services, allowing for the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. It now can be used to severe other models as well. To be modular and extensible the current specification itself is currently split into three complimentary documents:

- Core - this defines the OCCI model
- HTTP Rendering - this defines how to manipulate the core model using the OCCI RESTful API. The document defines how the OCCI model can be communicated and thus serialized using HTTP.
- Infrastructure - this defines the infrastructure domain resource types, the required attributes for each and the actions that can be taken on each.

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.

3 OCCI model

OCCI is a boundary protocol and API that acts as a service front-end to a provider's internal management framework. Figure 1 shows OCCI's place in a provider's architecture.

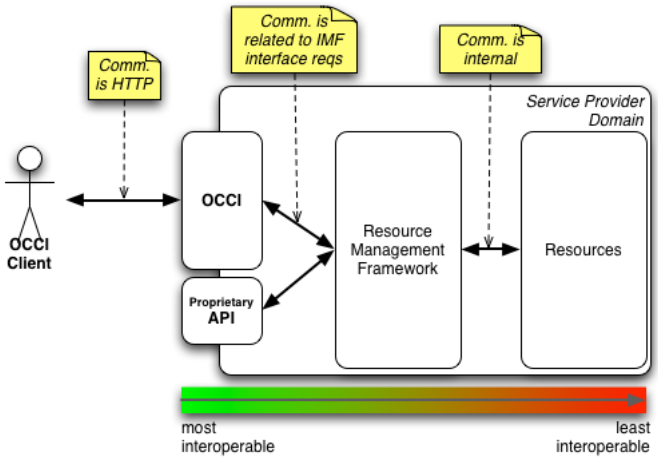

Figure 1. OCCI's place in a provider's architecture

The heart of the OCCI model is the `Resource` type. Any resource exposed through OCCI is a `Resource` or sub-type thereof. A resource can be e.g. a virtual machine, a job in a job submission system, a user, etc. The `Resource` type contains a number of common attributes that domain-specific `Resource` types inherit. The `Resource` type is complemented by the `Link` type which associates one `Resource` instance with another. The `Link` type also contains a number of common attributes that domain-specific `Link` types inherit.

`Kind` is an abstract type which both `Resource` and `Link` inherit. Each sub-type of `Kind` is identified by a unique `Type` instance. The `Type` type comprise the classification system built into the OCCI model. `Type` is a specialisation of `Category` and introduce additional capabilities in terms of `Action` types.

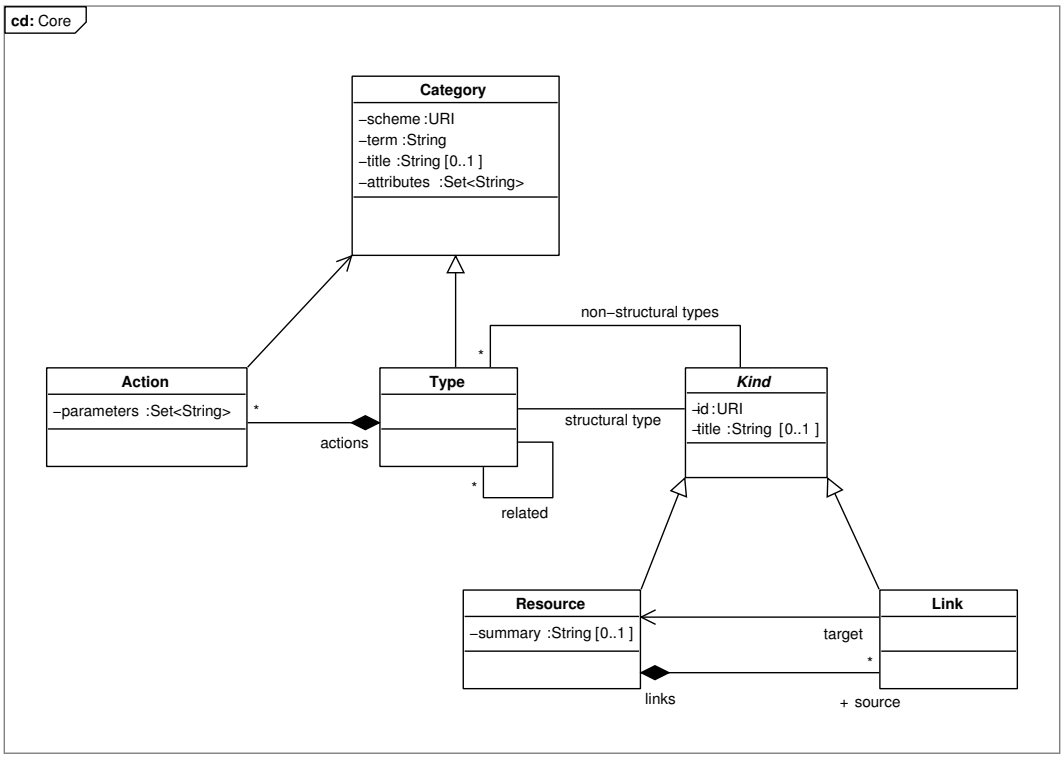

Figure 2. UML class diagram of the OCCI model. The diagram provides an overview of the OCCI model but is not a standalone definition thereof

The UML class diagram shown in figure 2 gives an overview of the OCCI model. For compliance with OCCI Core, all of the types defined in the OCCI model MUST be implemented. The following sections of the specification define the details of the OCCI model.

3.1 Classification and Identification

The OCCI model provides a built in classification system allowing for safe extension towards domain-specific usage. This system is like a "type system" but with the possibility of being easily exposed over a text based protocol. The classification system can be summarised with the following key features:

- Each OCCI base type and extension thereof is assigned a unique identifier, a structural `Type`, which allow for dynamic discovery of available types.
- The relationship of structural `Types` is part of the system and thus the inheritance model is also discoverable.
- The classification system allows non-structural `Types` to be assigned to resource instances adding new capabilities using a mix-in like model.
- Tagging of resource instances is supported through mix-in of non-structural `Types` which have no additional capabilities defined.
- A collection of associated resources is implicitly defined for each structural and non-structural `Type`. I.e. all resource instances associated with a particular `Type` instance form a collection.

3.1.1 Type

The `Type` type comprises the classification system provided by the OCCI model. It MUST be implemented. A `Type` instance can be either structural or non-structural.

Table 1. Attributes defined for the Type type

Attribute	Type	Multiplicity	Client Mutability	Description
actions	Action	0..*	Immutable	Set of Actions defined by the Type instance.
related	Type	0..*	Immutable	Set of related Type instances.
kind	Kind	0..1	Immutable	Kind type uniquely identified by the Type instance.

The Type type inherits the Category type and all inherited attributes MUST be implemented. Table 1 defines the additional attributes the Type type MUST implement to be compliant.

Structural Type A structural Type is an instance of Type assigned as the unique identifier of a Kind sub-type. The following rules apply:

- A structural Type define the capabilities of a Kind sub-type in terms of attributes and Actions.
- A unique structural Type MUST be assigned to each and every sub-type of Kind.
- A structural Type MUST be related, either directly or indirectly, to the structural Type of Kind, i.e. *http://schemas.ogf.org/occi/core#kind*. See section 3.1.3 for the definition of Type relationship.
- If type **B** inherit type **A**, where **A** is a sub-type of Kind, the structural Type of **B** MUST be directly related to the structural Type of **A**.

Non-structural Type A non-structural Type is an instance of Type *not* assigned as the unique identifier of any Kind sub-type. The following rules apply:

- A non-structural Type define additional capabilities for each Kind sub-type instance it is associated with. A non-structural Type add capabilities using a mix-in like model.
- A non-structural Type MUST only be associated with Kind sub-type *instances*, either at creation-time or run-time.
- A non-structural Type MUST NOT be related, neither directly nor indirectly, to the structural Type of Kind, i.e. *http://schemas.ogf.org/occi/core#kind*. See section 3.1.3 for the definition of Type relationship.
- A non-structural Type defining no additional capabilities in terms of attributes or Actions is considered to be a tag.

3.1.2 Category

The Category type comprises the basis of the identification mechanism used by the OCCI classification system. It MUST be implemented. Instances of the Category type are only used to identify Action types. All other uses of Category properties are managed through its sub-type Type. Table 2 defines the attributes the Category type MUST implement to be compliant.

Table 2. Attributes defined for the Category type

Attribute	Type	Multiplicity	Client Mutability	Description
term	String	1	Immutable	Unique identifier of the Category instance within the categorisation scheme.
scheme	URI	1	Immutable	The categorisation scheme.
title	String	0..1	Immutable	The display name of an instance.
attributes	String	0..*	Immutable	The set of resource attribute names defined by the Category instance.

A Category is uniquely identified by concatenating the categorisation scheme with the category term, e.g. *http://example.com/category/scheme#term*. This is done to enable discovery of Category definitions in text based renderings such as HTTP. All renderings MUST make use of and understand concatenated unique identifiers of Category types. Sub-types of Category such as Type inherit this property.

The categorisation schemes defined in the OCCI specification all use the *http://schemas.ogf.org/occi/* base URL. This base URL is reserved for OCCI an MUST NOT be used by domain-specific extensions.

Attribute names defined by Category instances[1] use the occi. prefix. This prefix is reserved for OCCI and MUST NOT be used by domain-specific extensions.

3.1.3 Type relationship

As previously defined a structural Type MUST be related, either either directly or indirectly, to the structural Type of Kind, i.e. *http://schemas.ogf.org/occi/core#kind*. The OCCI base types Resource and Link extend Kind. This together with any further sub-typing implies a hierarchy of related structural Type instances. The Type relationships thus mirror the type inheritance structure of the OCCI model and any extension thereof.

In an example where a domain-specific "Custom Compute Resource" is a sub-type the OCCI infrastructure type Compute, which in turn is a sub-type of the Resource type, four related structural Types would be involved. Table 3 illustrates the exemplified hierarchy of Type instances relating the domain-specific structural Type to the structural Type of Kind.

Table 3. Example of the Type relationship involved for a domain-specific extension of the OCCI infrastructure type Compute.

Structural Type	Related Structural Type
http://example.com/occi/custom#compute	*http://schemas.ogf.org/occi/infrastructure#compute*
http://schemas.ogf.org/occi/infrastructure#compute	*http://schemas.ogf.org/occi/core#resource*
http://schemas.ogf.org/occi/core#resource	*http://schemas.ogf.org/occi/core#kind*

3.1.4 Type assignment

A structural Type MUST be statically assigned to each sub-type of Kind defined by an implementation. A Kind sub-type instance MUST be automatically associated with its structural Type at creation-time. The structural Type associated with an instance MUST remain associated with the instance during its lifetime.

A non-structural Type, also known as a mix-in, MAY be associated with a Kind sub-type instance, either at creation-time or run-time. An OCCI implementation MAY restrict which instances can be associated with a particular non-structural Type.

3.1.5 Collections

One or more Kind sub-type instances associated with the same Type, may it be structural or non-structural, automatically form a collection. Each Type instance in the system identifies a collection consisting of all different Kind sub-type instances associated with the Type.

A Kind sub-type instance is always a member of the Kind's structural Type collection since a Kind sub-type instance MUST be associated with the structural Type of the Kind sub-type. Since a non-structural Type can be assigned to any Kind sub-type instance a collection can contain instances of different Kind sub-types. For example, an instance of the Resource type will always be associated to the structural Type *http://scheme.ogf.org/occi/core#resource* and thus part of the Resource Type collection.

Adding an instance to a collection is accomplished by associating the corresponding non-structural Type to the Kind sub-type instance.

[1] Also applies to Type instances.

Removing an instance from a collection is accomplished by disassociating the corresponding non-structural Type from the Kind sub-type instance.

An OCCI implementation MUST allow a client to navigate collections. The following basic navigation operations MUST be supported:

- Retrieve the whole collection.
- Retrieve a specific item in a collection.
- Retrieve a subset of a collection.

The details of collection navigation is rendering specific.

3.1.6 Discovery

An OCCI client MUST be able to discover all instances of Type and Category a particular service provider's OCCI implementation support. By examining these instances a client MUST be able to, at a minimum, deduce the following information:

- The Kind sub-types available from a the service provider, including domain-specific extensions.
- The attributes associated with each Kind sub-type.
- The invocable operations, i.e. Actions, defined for each Kind sub-type.
- Additional mix-ins or tags, i.e. non-structural Types, applicable to Kind sub-type instances.

The above requirements comprise the OCCI discovery mechanism. It MUST be implemented. The details of exactly how the Category and Type instances are exposed to an OCCI client is specific to the particular rendering used. The relevant details can be found in the OCCI rendering documents. References?

3.2 The OCCI base types

The following sections describe the OCCI base types defined by the OCCI model. The base types are Kind, Resource, Link and Action. All base types MUST be implemented.

3.2.1 Kind

The Kind type is the abstract base type for Resource and Link and any domain-specific sub-types thereof. It MUST be implemented. Table 4 defines the attributes the Kind type MUST implement to be compliant.

Table 4. Attributes defined for the Kind type.

Attribute	Type	Multiplicity	Client Mutability	Description
id	URI	1	Immutable	A unique identifier (within the service provider's namespace) of the Kind sub-type instance.
title	String	0..1	Mutable	The display name of the instance.
structural type	Type	1	Immutable	The structural Type of the instance.
non-structural types	Type	0..*	Mutable	The non-structural Types associated to this instance. Consumers can expect the attributes and Actions of the associated non-structural Types to be exposed by the instance.

Kind enforces for all sub-types a required id attribute and an optional title attribute. Every sub-type of Kind MUST be assigned a structural Type, see section 3.1.1. Kind itself is assigned the structural Type *http://schemas.ogf.org/occi/core#kind*. A Kind sub-type instance MAY be associated with one or more non-structural Types.

A Kind sub-type instance MUST expose its structural Type and any associated non-structural Types together with their associated attributes and Actions.

3.2.2 Resource

The Resource type inherit Kind and describes a concrete resource that can be inspected and manipulated. It represents a general object in the OCCI model and MUST be implemented. The Resource type MUST implement all attributes inherited from the Kind type together with the attributes defined in table 5 in order to be compliant. The Resource type is assigned the structural Type *http://schemas.ogf.org/occi/core#resource*.

Table 5. Attributes defined for the Resource type.

Attribute	Type	Multiplicity	Client Mutability	Description
summary	String	0..1	Mutable	A summarising description of the Resource instance.
links	Link	0..*	Mutable	A set of Link compositions. Being a composite relation the removal of a Link from the set MUST also remove the Link instance.

Resource enforces the inheritance of a set of common attributes into sub-types. Moreover, it introduces relationships to other Resource instances through instances of the Link type. The Resource type is the entry point for domain-specific extensions of the OCCI model, see section 3.4.

3.2.3 Link

An instance of the Link type defines a base association between two Resource instances. It MUST be implemented. A Link instance indicates that one Resource instance is connected to another. The Link type MUST implement all attributes inherited from the Kind type together with the attributes defined in table 6 in order to be compliant. The Link type is assigned the structural Type *http://schemas.ogf.org/occi/core#link*.

Table 6. Attributes defined for the Link type.

Attribute	Type	Multiplicity	Client Mutability	Description
source	Resource	1	Mutable	The Resource instances the Link instance originates from.
target	Resource	1	Mutable	The Resource instances the Link instance points to.

An instance of the Link type MUST NOT refer to an external resource. A provider MAY however create a sub-type of Link with different semantics, e.g. have a target attribute containing an URI and thus the ability of linking with external resources. The Link type can be sub-typed for domain-specific extensions of the OCCI model, see section 3.4.

3.2.4 Action

The Action type defines an invocable operation applicable to a Kind sub-type instance or a collection thereof. It MUST be implemented. In general, Actions modify state by e.g. performing a complex operation such as rebooting a virtual machine. Table 7 defines the attributes the Action type MUST implement to be compliant.

Table 7. Attributes defined for the Action type.

Attribute	Type	Multiplicity	Client Mutability	Description
category	Category	1	Immutable	The identifying Category of the Action.
parameters	String	0..*	Immutable	Enumeration of valid parameters for the Action.

An Action is always bound to a Type instance through a composite association. An Action is considered a capability of the Type. An Action MAY be invoked on any Kind sub-type instance associated with the Type instance defining the Action. An OCCI implementation MAY refuse an Action from being invoked if currently not applicable.

An Action MAY be invoked on a collection of Kind sub-type instances. The Action is only considered valid if all instances of the collection are associated with the Type defining the Action.

The Action type is assigned the Category identifier *http://schemas.ogf.org/occi/core#action*. The Action type can be sub-typed for domain-specific extensions of the OCCI model, see section 3.4.

3.3 Mutability

Attributes of an OCCI model type instance, a resource instance, are either client mutable or client immutable. If an attribute is noted to be mutable this MUST be interpreted that a client can create a resource instance that is parametrised by the attribute. Likewise, if an attribute is mutable, a client can update that resource instance's mutable attribute value and the server side MUST support this. If an attribute is marked as immutable, it indicates that the server side implementation MUST manage these exclusively. Immutable attributes MUST NOT be modifiable by clients under any circumstance.

3.4 Extensibility

The OCCI model has a flexible yet fairly simple extension mechanism based on the classification system described in section 3.1. The OCCI model can be extended using two different methods, sub-typing and mix-in. Both methods involve the use of domain-specific Type or Category instances. The following sections define the requirements for extensions of the OCCI model. The rules defined in section 3.1 and 3.2 are REQUIRED for all extensions of the OCCI model.

3.4.1 Type and Category instances

Domain-specific Type and Category instances MAY be introduced by an OCCI implementation. A Type or Category instance defined outside of the OCCI specification MUST use a categorisation scheme unique to the provider, e.g. *http://example.com/occi#*. An attribute introduced by a domain-specific Type or Category MUST use an attribute name prefix. This prefix MUST NOT be the "occi." prefix which is reserved for the OCCI specification. Domain-specific attribute names SHOULD use a prefix consisting of the provider's reverse domain name, e.g. "com.example.".

3.4.2 Sub-typing

The OCCI model MAY be extended through sub-typing for domain-specific purposes. Three OCCI model types MAY be sub-typed, those are Resource, Link and Action.

In order to define a sub-type of Resource or Link a domain-specific structural Type MUST be defined and assigned to the sub-type. This structural Type MUST be directly related to the structural Type of the type extended.

In order to define a sub-type of Action a domain-specific Category instance MUST be assigned to the Action sub-type as its unique identifier. Furthermore the Action sub-type MUST be associated as a capability of a domain-specific Type instance.

3.4.3 Mix-ins

The OCCI model MAY be extended through domain-specific mix-ins, i.e. non-structural Types. A non-structural Type MAY be associated with any Kind sub-type instance although a provider MAY apply restrictions.

In order to support user-defined tags an OCCI implementation must allow non-structural Types to be created and destroyed by request of a client. There is no limitation in the OCCI model from doing so but it is RECOMMENDED to assign a separate categorisation scheme for each user's non-structural Types[2].

[2] A tag is a non-structural Type which do not introduce additional capabilities.

4 Contributors

Editors: Andy Edmonds, Thijs Metsch, Ralf Nyrén
Contributors: Alexander Papaspyrou, Sam Johnston

TBD: Bunch op people missing here - create table...

5 Glossary

Term	Description
`Action`	An OCCI base type. Represent an invocable operation on a `Kind` sub-type instance or collection thereof.
`Category`	A type in the OCCI model. The parent type of `Type`.
Client	An OCCI client.
Collection	A set of `Kind` sub-type instances all associated to a particular `Type` instance.
`Kind`	An OCCI base type. The parent type of `Resource` and `Link`.
`Link`	An OCCI base type. A `Link` instance associate one `Resource` instance with another.
Mix-in	A non-structural `Type`.
Non-structural `Type`	An instance of `Type` *not* used as an unique identifier of an OCCI base type.
OCCI	Open Cloud Computing Interface
OCCI base type	One of `Kind`, `Resource`, `Link` or `Action`.
OGF	Open Grid Forum
`Resource`	An OCCI base type. The parent type for all domain-specific resource types.
Structural `Type`	An instance of `Type` assigned as the unique identifier of an OCCI base type.
Tag	A non-structural `Type` with no attributes or actions defined.
`Type`	A type in the OCCI model. The central piece in the OCCI classification system.

6 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.

7 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.

8 Full Copyright Notice

9 References

Note that only permanent documents should be cited as references. Other items, such as Web pages or working groups, should be cited inline (i.e., see the Open Grid Forum, http://www.ogf.org). References should conform to a standard such as used by IEEE/ACM, MLA, Chicago or similar. Include an author, year, title, publisher, place of publication. For online materials, also add a URL. It is acceptable to separate out "normative references," as IETF documents typically do. Some sample citations: