Draft
OCCI-WG

Thijs Metsch, Intel
Andy Edmonds, ICCLab, ZHAW
Boris Parák, CESNET

Updated: November 5, 2015

Open Cloud Computing Interface - Infrastructure

Status of this Document

This document is a draft including proposed errata updates to the OCCI Infrastructure [1] specification.

The errata updates are summarized in section A.

Eventually this document will obsolete GFD-P-R.143. This document is fully backward compatible to [1].

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

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 complimentary 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. The OCCI Core Model can be interacted through *renderings* (including associated behaviours) and expanded 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 inter-operable boundary interface between the web and the internal resource management system of infrastructure 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 [2].

[1] Infrastructure as a Service

3 Infrastructure

The OCCI Infrastructure document details how an OCCI implementation can model and implement an Infrastructure as a Service API offering by utilizing the OCCI Core Model. This API allows for the creation and management of typical resources associated with an IaaS service, for example, creating a Compute instance and Storage instance and then linking them with StorageLink. The main infrastructure types defined within OCCI Infrastructure are:

Compute Information processing resources.

Network Interconnection resource and represents a L2 networking resource. This is complimented by the IPNetwork Mixin.

Storage Information recording resources.

Supporting these Resource types are the following Link sub-types:

NetworkInterface connects a Compute instance to a Network instance. This complimented by an IPNetworkInterface Mixin.

StorageLink connects a Compute instance to a Storage instance.

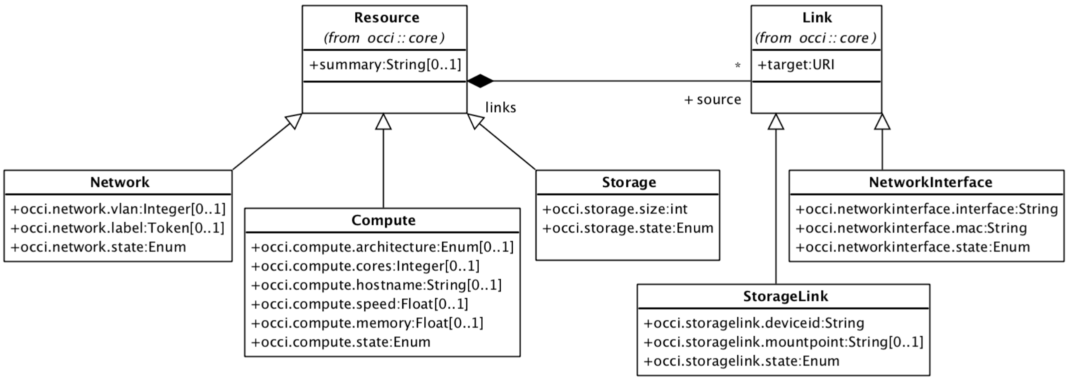

Resource
(from occi :: core)
+summary:String[0..1]
Link
(from occi :: core)
+target:URI
*
+ source
links
Network
+occi.network.vlan:Integer[0..1]
+occi.network.label:Token[0..1]
+occi.network.state:Enum
Compute
+occi.compute.architecture:Enum[0..1]
+occi.compute.cores:Integer[0..1]
+occi.compute.hostname:String[0..1]
+occi.compute.speed:Float[0..1]
+occi.compute.memory:Float[0..1]
+occi.compute.state:Enum
Storage
+occi.storage.size:int
+occi.storage.state:Enum
NetworkInterface
+occi.networkinterface.interface:String
+occi.networkinterface.mac:String
+occi.networkinterface.state:Enum
StorageLink
+occi.storagelink.deviceid:String
+occi.storagelink.mountpoint:String[0..1]
+occi.storagelink.state:Enum

Figure 1. Overview Diagram of OCCI Infrastructure Types.

These infrastructure types inherit the OCCI Core Model Resource base type and all their attributes. The HTTP Rendering document [3] defines how to serialize and interact with these types using RESTful communication. Implementers are free to choose what Resource and Link sub-types to implement. Those that are supported by an implementation will be discoverable through the OCCI Query Interface.

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 by setting the *parent* attribute. That assigned Kind instance MUST always remain immutable to any client.

Table 2. The Kind instances defined for the infrastructure sub-types of Resource, Link and related Mixins. The base URL **http://schemas.ogf.org/occi** has been replaced with <**schema**> in this table for a better readability experience.

Term	Scheme	Title	Parent Kind
compute	<schema>/infrastructure#	Compute Resource	<schema>/core#resource
storage	<schema>/infrastructure#	Storage Resource	<schema>/core#resource
storagelink	<schema>/infrastructure#	StorageLink Link	<schema>/core#link
network	<schema>/infrastructure#	Network Resource	<schema>/core#resource
networkinterface	<schema>/infrastructure#	NetworkInterface Link	<schema>/core#link

Table 2 describes the Kind instances defined for each of the infrastructure Resource or Link sub-types. For information on extending these types, please refer to the OCCI Core Model document [4].

The following sections on Compute, Storage and Network types detail the Attributes, Actions and states defined for each of them, including type-specific mixins where appropriate. Following those, the definition of infrastructure-related Link sub-types are given and finally OS and Resource Templates are defined. Figure 1 gives an overview of the key types involved in this infrastructure specification.

3.1 Compute

The Compute type represents a generic information processing resource, e.g. a virtual machine or container. Compute inherits the Resource base type defined in OCCI Core Model [4]. Compute is assigned the Kind instance *http://schemas.ogf.org/occi/infrastructure#compute*. A Compute instance MUST use and expose this Kind.

Table 3. Attributes defined for the Compute type.

Attribute	Type	Multi-plicity	Mutability	Description
occi.compute.architecture	Enum {x86, x64}	0. . . 1	Mutable	CPU Architecture of the instance.
occi.compute.cores	Integer	0. . . 1	Mutable	Number of virtual CPU cores assigned to the instance.
occi.compute.hostname	String	0. . . 1	Mutable	Fully Qualified DNS hostname for the instance.
occi.compute.share	Integer	0. . . 1	Mutable	Relative number of CPU shares for the instance.
occi.compute.memory	Float, 10^9 (GiB)	0. . . 1	Mutable	Maximum RAM in gigabytes allocated to the instance.
occi.compute.state	Enum {active, inactive, suspended, error}	1	Immutable	Current state of the instance.
occi.compute.state.message	String	0..1	Immutable	Human-readable explanation of the current instance state.

Table 3 describes the OCCI Attributes[2] defined by Compute through its Kind instance. These attributes MAY or MUST be exposed by an instance of the Compute type depending on the "Multiplicity" column in the aforementioned table.

Table 4. Actions applicable to instances of the Compute type. The Actions are defined by the Kind instance *http://schemas.ogf.org/occi/infrastructure#compute*. Every Action instance in the table uses the *http://schemas.ogf.org/occi/infrastructure/compute/action#* categorization scheme. "Action Term" below refers to Action.`term`.

Action Term	Target state	Attributes
start	active	–
stop	inactive	method={graceful, acpioff, poweroff}
restart	active (via stop and start chain)	method={graceful, warm, cold}
suspend	suspended	method={hibernate, suspend}
save	active (via stop and start chain)	method={hot, deferred}, name=*String*

Table 4 describes the Actions defined for Compute by its Kind instance. These Actions MUST be exposed by an instance of the Compute type of an OCCI implementation. Figure 2 illustrates the state diagram for a Compute instance.

Action "save" is expected to create an OS Template 3.5.1 referencing an independent copy of the current state of the Compute instance. The provider MAY choose to respect the "name" given by the client or override it according to its internal policies. A successful execution of this action MUST lead to a response containing the rendering of the newly created OS Template as defined by the chosen rendering and transport protocol. The

[2] See the "`attributes`" attribute defined by the Category type and inherited by Kind [4].

provider MAY choose to include a reference to the original Compute instance in `Mixin.Attributes` of the newly created OS Template.

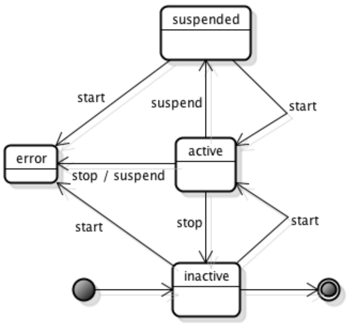


Figure 2. State Diagram for a Compute instance.

3.2 Network

The Network type represents a L2 networking entity (e.g. a virtual switch). It can be extended using the mixin mechanism (or sub-typed) to support L3/L4 capabilities such as TCP/IP etc. For the purposes of this specification we define an OCCI mixin so that IP networking can be supported where required. Network inherits the Resource base type defined in OCCI Core Model [4].

The Network type is assigned the *http://schemas.ogf.org/occi/infrastructure#network* Kind. A Network instance MUST use and expose this Kind.

Table 5. Attributes defined for the Network type.

Attribute	Type	Multi-plicity	Mutability	Description
occi.network.vlan	Integer: 0-4095	0. . . 1	Mutable	802.1q VLAN Identifier (e.g. 343).
occi.network.label	Token	0. . . 1	Mutable	Tag based VLANs (e.g. external-dmz).
occi.network.state	Enum {active, inactive, error}	1	Immutable	Current state of the instance.
occi.network.state.message	String	0..1	Immutable	Human-readable explanation of the current instance state.

Table 5 describes the OCCI Attributes[3] defined by Network through its Kind instance. These attributes MAY or MUST be exposed by an instance of the Network type depending on the "Multiplicity" column in the aforementioned table.

Table 6. Actions applicable to instances of the Network type. The Actions are defined by the Kind instance *http://schemas.ogf.org/occi/infrastructure#network*. Every Action instance in the table uses the *http://schemas.ogf.org/occi/infrastructure/network/action#* categorisation scheme. "Action Term" below refers to Action.`term`.

Action Term	Target State	Attributes
up	active	–
down	inactive	–

Table 6 describes the Actions defined for Network by its Kind instance. These Actions MUST be exposed by an instance of the Network type of an OCCI implementation. Figure 3 illustrates the state diagram for a Network instance.

[3] See the "`attributes`" attribute defined by the Category type and inherited by Kind [4].

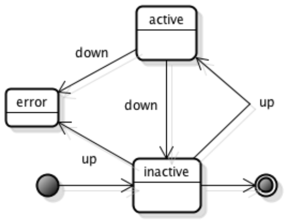


Figure 3. State Diagram for a Network instance.

3.2.1 IPNetwork Mixin

In order to support L3/L4 capabilities (e.g. IP, TCP etc.) an OCCI mixin is herewith defined.

The IPNetwork mixin is assigned[4] the "`scheme`" of *http://schemas.ogf.org/occi/infrastructure/network#* and the "`term`" value *ipnetwork*. An IPNetwork mixin MUST support these values.

Table 7 define the attributes introduced by the IPNetwork mixin.

The IPNetwork mixin MUST be related to the Network kind by setting the *applies* attribute to:

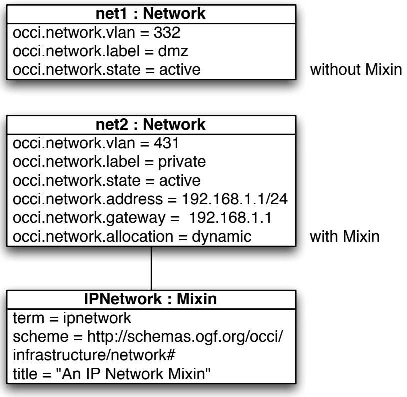
http://schemas.ogf.org/occi/infrastructure#network.

A Network instance associated with the IPNetwork mixin Mixin instance MUST implement these attributes.

Table 7. Attributes defined by the IPNetwork mixin. A Network instance associated with this Mixin instance MUST expose these attributes.

Attribute	Type	Multi-plicity	Mutability	Description
occi.network.address	IPv4 or IPv6 Address range, CIDR notation	0...1	Mutable	Internet Protocol(IP) network address (e.g. 192.168.0.1/24, fc00::/7)
occi.network.gateway	IPv4 or IPv6 Address	0...1	Mutable	Internet Protocol(IP) network address (e.g. 192.168.0.1, fc00::)
occi.network.allocation	Enum {dynamic, static}	0...1	Mutable	Address allocation mechanism: *dynamic* e.g. uses the dynamic host configuration protocol, *static* e.g. uses user supplied static network configurations.

In Figure 4 a UML object diagram depicts how Network would be associated with an IPNetwork Mixin when both are instantiated.



occi.network.vlan = 332
occi.network.label = dmz
occi.network.state = active

occi.network.vlan = 431
occi.network.label = private
occi.network.state = active
occi.network.address = 192.168.1.1/24
occi.network.gateway = 192.168.1.1
occi.network.allocation = dynamic

term = ipnetwork
scheme = http://schemas.ogf.org/occi/infrastructure/network#
title = "An IP Network Mixin"

Figure 4. Object Diagram of a Network Instance and its Associated IPNetwork Mixin.

[4] Both assignments use data members from the inherited Category type [4].

3.3 Storage

The Storage type represent resources that record information to a data storage device. Storage inherits the Resource base type defined in the OCCI Core Model [4]. The Storage type is assigned the Kind instance *http://schemas.ogf.org/occi/infrastructure#storage*. A Storage instance MUST use and expose this Kind.

Table 8. Attributes defined for the Storage type.

Attribute	Type	Multi-plicity	Mutability	Description
occi.storage.size	Float, 10^9 (GiB)	1	Mutable	Storage size in gigabytes of the instance.
occi.storage.state	Enum {online, offline, error}	1	Immutable	Current status of the instance.
occi.storage.state.message	String	0..1	Immutable	Human-readable explanation of the current instance state.

Table 8 describes the OCCI Attributes[5] defined by Storage through its Kind instance. These attributes MAY or MUST be exposed by an instance of the Storage type depending on the "Multiplicity" column in the aforementioned table.

Table 9. Actions applicable to instances of the Storage type. The Actions are defined by the Kind instance *http://schemas.ogf.org/occi/infrastructure#storage*. Every Action instance in the table uses the *http://schemas.ogf.org/occi/infrastructure/storage/action#* categorization scheme. "Action Term" below refers to Action.`term`.

Action Term	Target State	Attributes
online	online	–
offline	offline	–

Table 9 describes the Actions defined for Storage by its Kind instance. These Actions MUST be exposed by an instance of the Storage type of an OCCI implementation. Figure 5 illustrates the state diagram for a Storage instance.

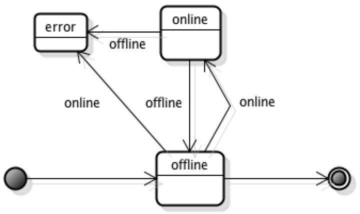


Figure 5. State Diagram for a Storage instance.

OCCI can be used in conjunction with the SNIA cloud storage standard, Cloud Data Management Interface (CDMI) [5] to provide enhanced management of the cloud computing storage and data. For storage managed through CDMI, see the section on StorageLink

3.4 Linking Infrastructure Resources

In order to create entities like virtual data centers or virtual clusters, it is necessary to allow the linkage of the previously defined infrastructure Resource sub-types. This is accomplished by extending (sub-typing) the OCCI Core Model Link base type. This is done as the Link base type cannot fully represent specific types of infrastructure links (e.g. links to storage or networks). These infrastructure links require additional attributes (e.g. network interface name) which can only be supported by sub-typing the Link base type.

[5] See the "`attributes`" attribute defined by the Category type and inherited by Kind [4].

3.4.1 Linking to Network

The NetworkInterface type represents an L2 client device (e.g. network adapter). It can be extended using the mix-in mechanism or sub-typed to support L3/L4 capabilities such as TCP/IP etc. NetworkInterface inherits the Link base type defined in the OCCI Core Model [4].

The NetworkInterface type is assigned the Kind instance *http://schemas.ogf.org/occi/infrastructure#networkinterface*. A NetworkInterface instance MUST use and expose this Kind. The Kind instance assigned to the NetworkInterface type MUST be related to the *http://schemas.ogf.org/occi/core#link* Kind by setting the *parent* attribute.

Table 10. Attributes defined for the NetworkInterface type.

Attribute	Type	Multiplicity	Mutability	Description
occi.networkinterface.interface	String	1	Immutable	Identifier that relates the link to the link's device interface
occi.networkinterface.mac	String	1	Mutable	MAC address associated with the link's device interface
occi.networkinterface.state	Enum {active, inactive, error}	1	Immutable	Current status of the instance.
occi.networkinterface.state.message	String	0..1	Immutable	Human-readable explanation of the current instance state.

Table 10 describes the OCCI Attributes[6] defined by NetworkInterface through its Kind instance. These attributes MAY or MUST be exposed by an instance of the NetworkInterface type depending on the "Multiplicity" column in the aforementioned table. Figure 6 illustrates the state diagram for a NetworkInterface instance.

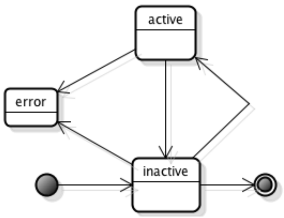


Figure 6. State Diagram for a NetworkInterface instance.

3.4.1.1 IPNetworkInterface Mixin In order to support L3/L4 capabilities (e.g. IP, TCP etc.) with the NetworkInterface type, an OCCI Mixin instance is herewith defined.

The IPNetworkInterface mixin is assigned[7] the "`scheme`" of *http://schemas.ogf.org/occi/infrastructure/networkinterface#* and the "`term`" value *ipnetworkinterface*. An IPNetworkInterface mixin MUST support these attributes.

The IPNetworkInterface mixin MUST be related to the NetworkInterface kind by setting the *applies* attribute to:

http://schemas.ogf.org/occi/infrastructure#networkinterface.

Table 11 define the attributes introduced by the IPNetworkInterface mixin. A NetworkInterface instance associated with the IPNetworkInterface mixin Mixin instance MUST expose these attributes.

In Figure 7 a UML object diagram depicts how NetworkInterface would be associated with an IPNetworkInterface Mixin when both are instantiated.

[6] See the "`attributes`" attribute defined by the Category type and inherited by Kind [4].

[7] Both assignments use data members from the inherited Category type [4].

Table 11. Attributes defined by the IPNetworkInterface mixin. A NetworkInterface instance associated with this Mixin instance MUST expose these attributes.

Attribute	Type	Multi-plicity	Mutability	Description
occi.networkinterface.address	IPv4 or IPv6 Address	1	Mutable	Internet Protocol(IP) network address (e.g. 192.168.0.1/24, fc00::/7) of the link
occi.networkinterface.gateway	IPv4 or IPv6 Address	0. . . 1	Mutable	Internet Protocol(IP) network address (e.g. 192.168.0.1/24, fc00::/7)
occi.networkinterface.allocation	Enum {dynamic, static}	1	Mutable	Address mechanism: *dynamic* e.g. uses the dynamic host configuration protocol, *static* e.g. uses user supplied static network configurations.

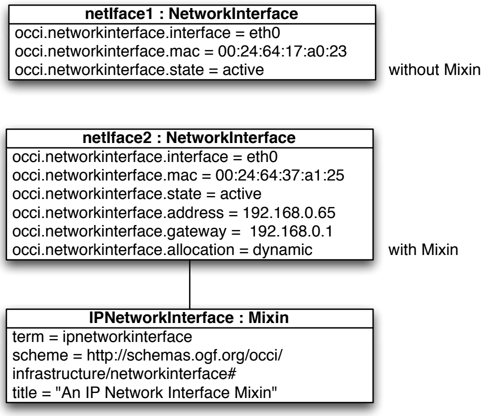

netIface1 : NetworkInterface
occi.networkinterface.interface = eth0
occi.networkinterface.mac = 00:24:64:17:a0:23
occi.networkinterface.state = active
without Mixin
netIface2 : NetworkInterface
occi.networkinterface.interface = eth0
occi.networkinterface.mac = 00:24:64:37:a1:25
occi.networkinterface.state = active
occi.networkinterface.address = 192.168.0.65
occi.networkinterface.gateway = 192.168.0.1
occi.networkinterface.allocation = dynamic
with Mixin
IPNetworkInterface : Mixin
term = ipnetworkinterface
scheme = http://schemas.ogf.org/occi/
infrastructure/networkinterface#
title = "An IP Network Interface Mixin"

Figure 7. Object Diagram of a NetworkInterface Instance and its Associated IPNetworkInterface Mixin.

3.4.2 Linking to Storage

The StorageLink type represents a link from a Resource to a target Storage instance. This enables a Storage instance be attached to a Compute instance, with all the prerequisite low- level operations handled by the OCCI implementation. This mechanism SHOULD NOT be used to choose an operating system for the given Compute instance, see Section 3.5.1. Storage inherits the Link base type defined in the OCCI Core Model [4].

The StorageLink type is assigned the Kind instance *http://schemas.ogf.org/occi/infrastructure#storagelink*. A StorageLink instance MUST use and expose this Kind. The Kind instance assigned to the StorageLink type MUST be related to the *http://schemas.ogf.org/occi/core#link* Kind by setting the *parent* attribute.

Table 12. Attributes defined for the StorageLink type.

Attribute	Type	Multi-plicity	Mutability	Description
occi.storagelink.deviceid	String	1	Mutable	Device identifier as defined by the OCCI service provider.
occi.storagelink.mountpoint	String	0. . . 1	Mutable	Point to where the storage is mounted in the guest OS.
occi.storagelink.state	Enum {active, inactive, error}	1	Immutable	Current status of the instance.
occi.storagelink.state.message	String	0..1	Immutable	Human-readable explanation of the current instance state.

Table 12 describes the OCCI Attributes[8] defined by StorageLink through its Kind instance. These attributes MAY or MUST be exposed by an instance of the StorageLink type depending on the "Multiplicity" column in the aforementioned table. Figure 8 illustrates the state diagram for a StorageLink instance.

[8] See the "`attributes`" attribute defined by the Category type and inherited by Kind [4].

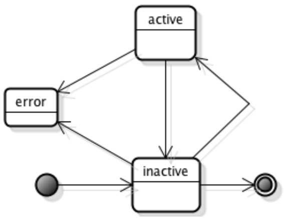


Figure 8. State Diagram for a StorageLink instance.

3.4.3 Linking to CDMI Managed Storage

As previously stated, OCCI can be used in conjunction with the SNIA cloud storage standard, Cloud Data Management Interface (CDMI) [5] to provide enhanced management of the cloud computing storage and data. In order to integrate the two, the use of StorageLink should be used. This will link OCCI managed Resources to CDMI resources. The 'occi.storagelink.deviceid' attribute of StorageLink, defined above, should be set to the CDMI Object ID of an exported CDMI Container.

3.5 Infrastructure Templates

Infrastructure Templates allow clients of an OCCI implementation to quickly and conveniently apply pre-defined configurations to OCCI Infrastructure defined types. They are implemented using Mixin instances. There are 2 supported infrastructure template types in OCCI Infrastructure.

3.5.1 OS Template

OS (Operating System) Templates allow clients specific what operating system must be installed on a requested Compute resource. OCCI implementations SHOULD support this, otherwise what they provision will be merely offer Resources without any available execution environment (e.g. operating system). Of the two supported template types, this is the most basic and necessary template that a provider SHOULD offer.

Its construction is a Mixin instance consisting of a provider specific "scheme" and a descriptive "title" detailing the OS. The "term" value of the template Mixin is a provider-specific identifier that corresponds to a particular image configuration. Where an implementation requires additional attributes associated with the OS Template, it can do so using "`attributes`" value inherited from the Category type.

Default values for OCCI Attributes defined by the Kind or the OS Template Mixin MAY be provided using the Attribute.`default` attribute property [4].

An implementation-defined OS Template Mixin MUST be related to the OCCI OS Template Mixin in order to give absolute type information by setting the *depends* attribute.

The OCCI OS Template is defined by the *http://schemas.ogf.org/occi/infrastructure#os_tpl* Mixin and MUST be supported SHOULD OS Templates be offered by the OCCI implementation.

If an OS Template is already associated with the given Resource instance, associating a new OS Template (using mechanisms defined by the chosen rendering and transport protocol) MUST result in an immediate removal of the old OS Template and association of the new OS Template. The change must affect the operating system of the given Resource instance, in a provider-specific way.

A typical example of using such a Mixin is shown in figure 9 using a UML object diagram. In the example illustrated in figure 9 a provider has defined an OS template which offers the ability to run Ubuntu Linux, version 9.10, upon a client's provisioned compute resource.

How a provider manages their set of OS templates will be determined by the provider and implementation-specific.

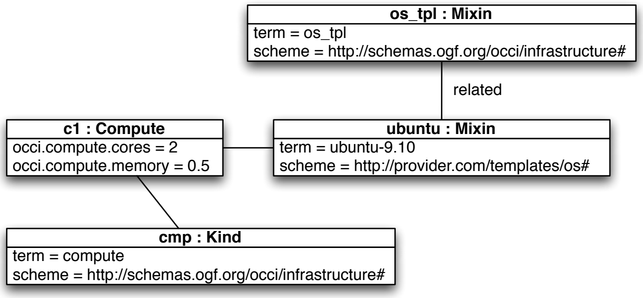

os_tpl : Mixin
term = os_tpl
scheme = http://schemas.ogf.org/occi/infrastructure#
related
c1 : Compute
occi.compute.cores = 2
occi.compute.memory = 0.5
ubuntu : Mixin
term = ubuntu-9.10
scheme = http://provider.com/templates/os#
cmp : Kind
term = compute
scheme = http://schemas.ogf.org/occi/infrastructure#

Figure 9. Object Diagram of a Compute Instance and its Associated OS Template Mixin.

3.5.2 Resource Template

The Resource Template Mixin builds upon the concept of OS Templates. A Resource Template is a provider-defined Mixin instance that refers to a pre-set Resource configuration. If a Resource Template Mixin is not provided, the provider is free to choose a default pre-set Resource configuration. If a Resource instance carries its own "size"-related attributes, an assigned Resource Template Mixin will override them where applicable.

The pre-set Resource configuration is not fully visible through the OCCI Discovery mechanism, depending on the chosen OCCI rendering and necessary provider-specific implementation details. The Mixin.`attributes` (inherited from Category) for a Resource Template Mixin SHOULD contain relevant attributes and default attribute values. Provider-specific side-effects are handled by the implementation and MUST not be exposed.

The OCCI implementation associates a set of Resource attributes (via Category's "attributes") with a particular term identifier.

An implementation-defined Resource Template Mixin MUST be related to the OCCI Resource Template Mixin in order to give absolute type information. This is done by setting the *depends* attribute. The OCCI Resource Template is defined by the Mixin instance *http://schemas.ogf.org/occi/infrastructure#resource_tpl* and MUST be supported SHOULD Resource Templates be offered by the OCCI implementation.

If a Resource Template is already associated with the given Resource instance, associating a new Resource Template (using mechanisms defined by the chosen rendering and transport protocol) MUST result in an immediate removal of the old Resource Template and association of the new Resource Template. The change must affect the the given Resource instance, in a provider-specific way (e.g., resizing the instance).

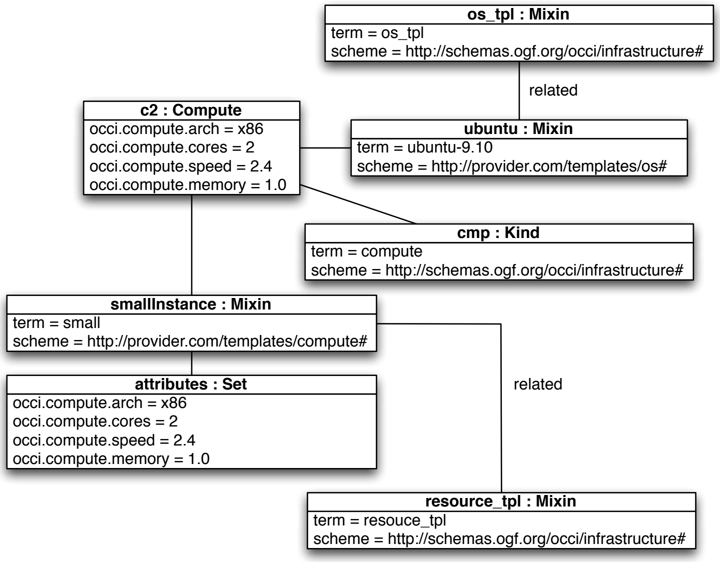


Figure 10. Object Diagram of a Compute Instance and its Associated OS Template Mixin and Resource Template Mixin.

A typical example of such a Mixin's use is shown in figure 10) using a UML object diagram. In this example, the provider offers Compute Resources based on different sizes (i.e. small, medium, large). Each "size" of Compute (i.e. the term) corresponds to a predetermined set of OCCI Resource-specific attributes. In the example below a "small" Compute instance is created. Specifying "small" as the term corresponds to an implementation-specific Compute Resource-specific attribute set that is shown by the object instance named "attributes" in figure 10. When this Mixin is associated with a Compute instance, the Compute instance will take on provided attributes and default attribute values.

From the administrative point of view, how an OCCI service provider manages their set of Resource Templates will be determined by the provider and so is implementation-specific.

3.5.2.1 Credentials Mixin When creating a Compute Resource a client normally supplies security credentials in the form of a public SSH key. This SSH key is injected into the Compute Resource by the provider on the client's behalf. This feature is provided by the Credentials Mixin.

If a provider that offers VMs with access secured by SSH then that OCCI implementation SHOULD support this. Otherwise no user supplied public SSH key can be injected into the Compute Resource.

The OCCI credentials mixin has the term *'ssh_key'* and the schema *'http://schemas.ogf.org/occi/credentials#'*.

The credentials mixin MUST only apply to the Compute Kind and therefore the mixin should have its *applies* attribute set to:

http://schemas.ogf.org/occi/infrastructure#compute.

Table 13. Attributes defined by the Credentials mixin. A Compute instance associated with this Mixin instance MUST expose these attributes.

Attribute	Type	Multi-plicity	Mutability	Description
occi.credentials.ssh.publickey	String	1	Mutable	The contents of the public key file to be injected into the Compute Resource

3.5.2.2 Contextualization Mixin In order to ease automation, OCCI supports the means to execute a program once Resource has been instantiated. This feature is provided by the contextualization mixin. On receipt of the contextualization data the OCCI implementation MUST distinguish the type of data being presented and then supply that content to the Compute Resource being instantiated. That content is then executed by the Compute Resource as the last step in the Compute's boot-order.

OCCI implementations SHOULD support this otherwise no contextualization of a resource instance can be done. The OCCI contextualization mixin has the term *user_data* and the schema *http://schemas.ogf.org/occi/compute#*

Contextualization mixin MUST only apply to the Compute Kind and therefore the mixin should have its *applies* attribute set to:

http://schemas.ogf.org/occi/infrastructure#compute.

Table 14. Attributes defined by the Contextualization mixin. A Compute instance associated with this Mixin instance MUST expose these attributes.

Attribute	Type	Multi-plicity	Mutability	Description
occi.compute.userdata	String	1	Mutable	Contextualization data (e.g. script, executable) that the client supplies once and only once. It cannot be updated.

4 Security Considerations

The OCCI Infrastructure specification is an extension to the OCCI Core and Model specification [4]; thus the same security considerations as for the OCCI Core and Model specification apply here.

5 Glossary

Term	Description
Action	An OCCI base type. Represents an invocable operation on a 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.

6 Contributors

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

Name	Affiliation	Contact
Michael Behrens	R2AD	behrens.cloud at r2ad.com
Mark Carlson	Toshiba	mark at carlson.net
Augusto Ciuffoletti	University of Pisa	augusto.ciuffoletti at gmail.com
Andy Edmonds	ICCLab, ZHAW	edmo at zhaw.ch
Sam Johnston	Google	samj at samj.net
Gary Mazzaferro	Independent	garymazzaferro at gmail.com
Thijs Metsch	Intel	thijs.metsch at intel.com
Ralf Nyrén	Independent	ralf at nyren.net
Alexander Papaspyrou	Adesso	alexander at papaspyrou.name
Boris Parák	CESNET	parak at cesnet.cz
Alexis Richardson	Weaveworks	alexis.richardson at gmail.com
Shlomo Swidler	Orchestratus	shlomo.swidler at orchestratus.com
Florian Feldhaus	NetApp	florian.feldhaus at gmail.com

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

7 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.

8 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.

9 Full Copyright Notice

References

[1] T. Metsch and A. Edmonds, "Open Cloud Computing Interface – Infrastructure," GFD-P-R.184, April 2011. [Online]. Available: http://ogf.org/documents/GFD.184.pdf

[2] 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

[3] T. Metsch and A. Edmonds, "Open Cloud Computing Interface – HTTP Rendering," GFD-P-R.185, April 2011. [Online]. Available: http://ogf.org/documents/GFD.185.pdf

[4] R. Nyrén, A. Edmonds, A. Papaspyrou, and T. Metsch, "Open Cloud Computing Interface – Core," GFD-P-R.183, April 2011. [Online]. Available: http://ogf.org/documents/GFD.183.pdf

[5] D. Slik, M. Siefer, E. Hibbard, C. Schwarzer, A. Yoder, L. N. Bairavasundaram, S. Baker, M. Carlson, H. Nguyen, and R. Ramos, "Cloud data management interface (cdmi) v1.0," http://www.snia.org/, Apr. 2010. [Online]. Available: http://www.snia.org/tech_activities/standards/curr_standards/cdmi/CDMI_SNIA_Architecture_v1.0.pdf

A Errata

- New credentials mixin - allows credentials to be supplied to the creation of a compute resource
- New contextualization mixin - allows a script to be supplied with the creation request of a compute resource
- Added error state to all resource state models
- Added occi.compute.share attribute to Compute. This allows for basic support of container virtualization technologies.
- Added state.message to all infrastructure resources (Compute, Storage, Network, NetworkInterface, StorageLink)
- Added references to the core model parent, applies and depends for infrastructure mixins, kinds
- Updated figures to reflect new Core model
- Updated the storage state model - removes resize. removal of error action from tables. resize done through a resource update
- Removed backup, snapshot, resize and degraded actions from state tables