GFD-I
OCCI-WG

Augusto Ciuffoletti, Università di Pisa

September 22, 2014
Updated: April 13, 2015

Open Cloud Computing Interface - Monitoring Extension

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.

This document *defines* an OCCI Extension that allows the user to create a monitoring infrastructure for a cloud provision. The Monitoring Extension *introduces* two further entity types: the *Collector* Link, that performs measurements, and the *Sensor* Resource, that processes them.

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 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 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. They do not require changes to the HTTP Rendering specifications as of this version of the specification.

The current specification consists of three documents. This specification describes version 1.1 of OCCI. Future releases of OCCI may include additional rendering and extension specifications. The documents of the current OCCI specification suite are:

OCCI Core describes the formal definition of the the OCCI Core Model [1].

OCCI HTTP Rendering defines how to interact with the OCCI Core Model using the RESTful OCCI API [2]. The document defines how the OCCI Core Model can be communicated and thus serialised using the HTTP protocol.

OCCI Infrastructure contains the definition of the OCCI Infrastructure extension for the IaaS domain [3]. The document extends the OCCI Core Model with additional Entity sub-types and their associated attributes and actions.

2 Features of the OCCI Monitoring Extension

A cloud monitoring infrastructure collects quantitative measurements of the performance of the provision; the results are used to implement relevant services like autonomic management, billing, and service level verification. This document defines an *OCCI Extension* that provides the user with the an interface to request the creation of a cloud monitoring infrastructure. The interest for a standard interface is that it enables the user to manage distinct cloud providers, possibly at the same time, in a uniform way.

The *OCCI Monitoring Extension* is extremely flexible regarding the description of the management of the flow of measurements, to accomodate a variated and growing range of applications: it enables the user to describe the task, but leaves the provider the the task of implementing a solution fitting the terms of the specific case.

The metrics that are used to evaluate the performance of a cloud provision are many, and subject to continuous changes due to the introduction of new technologies and services. It is therefore impossible to give a *detailed* framework that extends its validity to any conceivable use case or provider. The OCCI Monitoring Extension

[1] Infrastructure as a Service

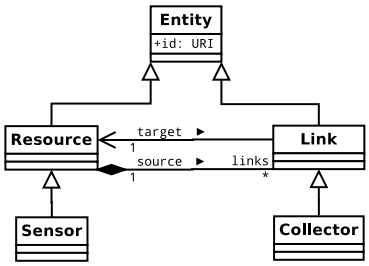

Figure 1. Class inheritance diagram for OCCI-monitoring

allows the user to describe the components a monitoring infrastructure, but does not interfere with details that may be transparently dealt with by the provider. Its application is not limited to the realm of computing infrastructures, since it is not bound to definite kinds of resources.

The OCCI Monitoring Extension is designed to scale from very simple use cases, that the user can define with minimal effort, to complex cases, like multilayer monitoring infrastructures. The same basic building blocks are used at any degree of complexity.

The OCCI Monitoring Extension defines two new entity types: the *Collector*, an OCCI-Link that defines the timing of the measurement process, and the *Sensor*, an OCCI-Resource that defines how the results of measurements are processed and used. The user defines the frequency with which measurements are collected and the time lapse during which the measurement process takes place.

Such a *real-time* approach excludes those cases that envision an *event driven* notification of a state change: the OCCI Notification Extension covers this case and is complementary to *OCCI monitoring*.

To specify the details of the monitoring activity the user associates provider-specific *mixin*s to a *Sensor* or *Collector* instance. Three tags are introduced to classify the *mixin*s: the *Metric* tag is associated with mixins that specify the production of measurement, the *Aggregator* tag is associated with mixins that describe how measurements are processed, and the *Publisher* tag is associated with mixins that control how results are published. Provider specific *Mixin*s are *related* to such tags, so to enforce a kind of subtyping.

3 Specification of the server that provides OCCI Monitoring

The compliant server MUST implement the *kind*s in table 1, and the *mixin*s in table 2.

Table 1. The immutable model attributes of the *Sensor* and of the *Collector* kinds. The base URL **http://schemas.ogf.org/occi** has been replaced with <**schema**> in this table for a better readability experience.

Term	Scheme	Title	Attributes	Actions	Parent
sensor	<schema>/monitoring#	Sensor Resource	see Table 3	{}	<schema>/core#resource
collector	<schema>/monitoring#	Collector Link	see Table 4	{}	<schema>/core#link

The attributes of the two kinds are illustrated in section 3.1 and in section 3.2 respectively.

The mixins have no capabilities: they are used as tags for other mixins with given features, as described in section 3.3, 3.4, 3.5.

Table 2. The immutable model attributes of the *Metric*, *Aggregator* and *Publisher* mixins. The base URL **http://schemas.ogf.org/occi** has been replaced with <**schema**> in this table for a better reading experience.

Term	Scheme	Title	Attributes	Actions	Depends	Applies
metric	<schema>/monitoring#	Metric Mixin	{}	{}	{}	<schema>/monitoring#collector
aggregator	<schema>/monitoring#	Aggregator Mixin	{}	{}	{}	<schema>/monitoring#sensor
publisher	<schema>/monitoring#	Publisher Mixin	{}	{}	{}	<schema>/monitoring#sensor

3.1 The Sensor resource

Table 3. Attributes of the Sensor resource.

Attribute	Type	Multi-plicity	Mutability	Description
occi.sensor.timebase	string	0..1	true	Base time reference (ISO8601)
occi.sensor.timestart	number	0..1	true	Start time offset (seconds)
occi.sensor.timestop	number	0..1	true	Stop time offset (seconds)
occi.sensor.period	number	1	true	Time between two following measurements (seconds)
occi.sensor.granularity	number	0..1	true	Granularity of time measument (seconds)
occi.sensor.accuracy	number	0..1	true	Accuracy of time measument (seconds)

The *kind* instance assigned to the *Sensor* type is `http://schemas.ogf.org/occi/monitoring#sensor`, as in table 1. The attributes of the *Sensor* (see table 3) describe the real-time properties of the sequence of metric values: the time interval during which the monitoring activity is in effect, and its frequency.

The start of the monitoring activity is determined as the sum between a reference date and time (`timebase`) and an offset (`timestart`). Another offset determines the time when the monitoring activity terminates (`timestop`). The attributes may be left undefined, to indicate that the monitoring activity persists for the lifetime of the sensor. A negative `timebase` indicates that the monitoring activity is suspended. The three attributes can be modified by the user to dynamically control the monitoring activity.

The frequency of the monitoring operation is controlled by a value that corresponds to the time lapse between successive samples (`period`). This attribute is required.

The precision of the time scale is defined by two attributes, the `granularity` and the `accuracy`. The former represents the minimum distance in time between two following time values, the latter indicates the maximum distance between the measured time and the real time. These attributes may be left unspecified.

The server SHOULD respond with an error, without allocating the *Sensor* resource, when the period of operation is partially in the past, or when the stop time precedes the start time. The server SHOULD NOT respond with an error when the `period`, the `granularity` or the `accuracy` cannot be met. Instead, it SHOULD attempt a *best fit* of the requested values, setting the appropriate values for the attributes. The server SHOULD fill undefined `granularity` or the `accuracy` attributes with worst case values.

3.2 The Collector link

Table 4. Attributes of the Collector link.

Attribute	Type	Multi-plicity	Mutability	Description
occi.collector.period	number	1	true	Time between two following measurements (seconds)
occi.collector.granularity	number	0..1	true	Granularity of time measument (seconds)
occi.collector.accuracy	number	0..1	true	Accuracy of time measument (second

The *kind* instance assigned to the *Collector* type is `http://schemas.ogf.org/occi/monitoring#collector`, as in table 1. The attributes of the *Collector* (see table 4) model the activity that extracts measurements from a destination *resource* and the transfer of such measurements to a source *Sensor*. The *source* of a *Collector* MUST be a *Sensor*: the server MUST respond with an error to the request of instantiation of a *Collector* if the source is not a *Sensor*.

The OCCI attributes of the *Collector* define the timing of the monitoring activity. The execution rate is defined using three attributes: the rate itself (`period`), and the quality of the timing (`granularity` and `timing`). All three values can be left unspecified.

The server SHOULD NOT respond with an error when the `period`, the `granularity` or the `accuracy` cannot be met. Instead, it SHOULD attempt a *best fit* of the requested values, setting the appropriate values

for the attributes. The server SHOULD fill undefined `granularity` or `accuracy` attributes with worst case values.

3.3 Features of the mixins that depend on the Metric tag

The *mixin* instance assigned to the *metric* mixin is `http://schemas.ogf.org/occi/monitoring#metric`, as in table 2. The *metric* mixin has no capabilities, and is used as a tag for mixins that customize a *Collector* link, by specifing the collected metrics and the measurement process.

The OCCI attributes of a *mixin* that depends on the *metric* tag are divided into two groups:

- Metric attributes: they represent the delivered measurements. They are *String* identifiers that are associated with *output channel*s (see section 3.6);
- Control attributes: they control the operation of the measurement activity.

3.4 Features of the mixins that depend on the Aggregator tag

The *mixin* instance assigned to the *aggregator* mixin is `http://schemas.ogf.org/occi/monitoring#aggregator`, as in table 2. The *aggregator* mixin has no capabilities, and is used as a tag for mixins that customize a *Sensor* resource, by specifing how raw metrics are processed before being published.

The OCCI attributes of a *mixin* that depends on the *aggregator* tag are divided into three groups:

- Input attributes: they bind an input of the aggregating algorithm with the metrics coming from monitored resources connected with outgoing *Collector*s. They are *String* identifiers that are associated with *input channel*s (see section 3.6);
- Control attributes: they control the operation of the aggregating function;
- Metric attributes: they represent the delivered measurements. They are *String* identifiers that are associated with *output channel*s (see section 3.6).

3.5 Features of the mixins that depend on the Publisher tag

The *mixin* instance assigned to the *publisher* mixin is `http://schemas.ogf.org/occi/monitoring#publisher`, as in table 2. The *publisher* mixin has no capabilities, and is used as a tag for mixins that customize a *Sensor* resource, by specifing how metrics are published.

The OCCI attributes of a *mixin* that depends on the *publisher* tag are divided into two groups:

- Input attributes: they bind the publishing process with the metrics produced by metric or aggregator mixins. They are *String* identifiers that are associated with an *input channel* (see section 3.6);
- Control attributes: they control the process used to publish input attributes.

3.6 The scope and the monitoring pipe

The three types of mixins — namely, *metric*, *aggregator*, and *publisher* — are meant to represent the three stages of a pipe. Measurements flow across the three stages, from the monitored resource to the point where they are published. The provider has control on the monitoring infrastructure the user is able create, since this ultimately depend on the available *mixin*s of the above types.

In case the provider allows the user to compose the building blocks, instead of offering only pre-configured monitoring schemas, the API must be powerful enough to define the flow of data between the building blocks. Input and output *channel* attributes fit the purpose: when they share the same value (or *channel* identifier), the measurements flow from input ones to output ones.

The visibility or *scope* of *channel* identifiers is limited within a *Sensor* and its outgoing *Collector*.

According with the core properties of *resources* and *links* (see [1]), the removal of a *Sensor* determines the removal of all *Collector*s in its *scope*.

4 Conformance profiles

The definition of conformance profiles is appropriate because the provision of an extension is optional, so it is appropriate to define a conformant behavior when the extension is not implemented, or when it is implemented at a lesser degree.

Profile 0 The *Collector* and *Sensor Kind* s MUST NOT be implemented: attempt of instantiating entities of such *Kinds* fails. The *Aggregator*, *Metric*, and *Publisher mixin*s MUST NOT be implemented: their discovery fails;

Profile 1 The *Collector* and *Sensor Kind* s MUST be implemented, and the user MUST be allowed to create new instances of such *Kinds*. The *Aggregator*, *Metric*, and *Publisher mixin* MUST be implemented, and discovery is successful. The server MUST NOT allow to introduce *depends* relationships with the *Aggregator*, *Metric*, and *Publisher mixin*s;

Profile 2 The *Collector* and *Sensor Kind* s MUST be implemented, and the user MUST be allowed to create new instances of such *Kinds*. The *aggregator*, *metric*, and *publisher mixin*s MUST be implemented, and discovery is successful. The user MUST be allowed to introduce *depends* relationships with *Aggregator*, *Metric*, and *Publisher mixin*s.

5 Security issues

The OCCI Notification specification is an extension to the OCCI Core and Model specification [1]; thus the same security considerations as for the OCCI Core and Model specification apply here.

6 Glossary

Term	Description
Action	An OCCI base type. Represent 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 OCCI Attributes and OCCI Actions exposed by an **entity instance**.
Client	An OCCI client.
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 associate 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.
model attribute	An internal attribute of a the Core Model which is *not* client discoverable.
OCCI	Open Cloud Computing Interface.
OCCI base type	One of Entity, Resource, Link or Action.
OCCI Action	see Action.
OCCI Attribute	A client discoverable attribute identified by an instance of the Attribute type. Examples are occi.core.title and occi.core.summary.
OCCI Category	see Category.
OCCI Entity	see Entity.
OCCI Kind	see Kind.
OCCI Link	see Link.
OCCI Mixin	see Mixin.
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.
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

Augusto Ciuffoletti (corresponding author)
Dept. of Computer Science

L.go B. Pontecorvo - Pisa
Italy
Email: augusto.ciuffoletti@gmail.com

Andrew Edmonds
Institute of Information Technology
Zürich University of Applied Sciences
Zürich
Switzerland
Email: andrew.edmonds@zhaw.ch

Metsch, Thijs
Intel Ireland Limited
Collinstown Industrial Park
Leixlip, County Kildare, Ireland Email: thijsx.metsch@intel.com

Ralf Nyren
Email: ralf@nyren.net

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

References

[1] 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

[2] 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

[3] ——, "Open Cloud Computing Interface – Infrastructure," GFD-P-R.184, April 2011. [Online]. Available: http://ogf.org/documents/GFD.184.pdf