Draft
OCCI-WG

Gregory Katsaros, Intel

December 15, 2014

Open Cloud Computing Interface - Service Level Agreements

Status of this Document

This document is a draft providing information to the community regarding the specification of the Open Cloud Computing Interface.

Copyright Notice

Trademarks

Abstract

This document, part of a document series, produced by the OCCI working group within the Open Grid Forum (OGF), provides a high-level definition of a Protocol and API in relation with the Service Level Agreements extension of the OCCI Core Model. 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 3

2 Notational Conventions 3

3 Service Level Agreement 3

3.1 Agreement . . . 4

3.1.1 AgreementTemplate Mixin . . . 6

3.1.2 AgreementTerm Mixin . . . 6

3.2 AgreementLink . . . 8

3.3 OCCI Service Level Agreement example . . . 8

4 Security Considerations 9

5 Glossary 10

6 Contributors 11

7 Intellectual Property Statement 11

8 Disclaimer 11

9 Full Copyright Notice 11

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 through *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.

TODO: replace with 1.2, note backwards compatibility. define new set of docs for 1.2 below...

2 Notational Conventions

All these parts and the information within are mandatory for implementors (unless otherwise specified). The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

3 Service Level Agreement

The OCCI Service Level Agreements (OCCI SLAs) document describes how the OCCI Core Model [2] can be extended and used to implement a Service Level Agreement management API. This API allows for the creation and management of resources related with the realization of agreements between an OCCI-enabled cloud service provider and potential consumers of the provider's resources. The introduced types and Mixins defined in this OCCI SLAs document are the following:

Agreement This resource represents the Service Level Agreement between the provider and the consumer. It includes the basic information for this contract and with the appropriate extensions (Mixins) it can be populated with further information. To this end, we introduce the AgreementTemplate and the AgreementTerms Mixins which complement the SLAs with template tagging and terms specification respectively.

AgreementLink This is a link entity that associates an Agreement instance with any other Resource instance.

These infrastructure types inherit the OCCI Core Model Resource base type and all their attributes. The HTTP Rendering document [3] defines how to serialise and interact with these types using RESTful communication.

[1] Infrastructure as a Service

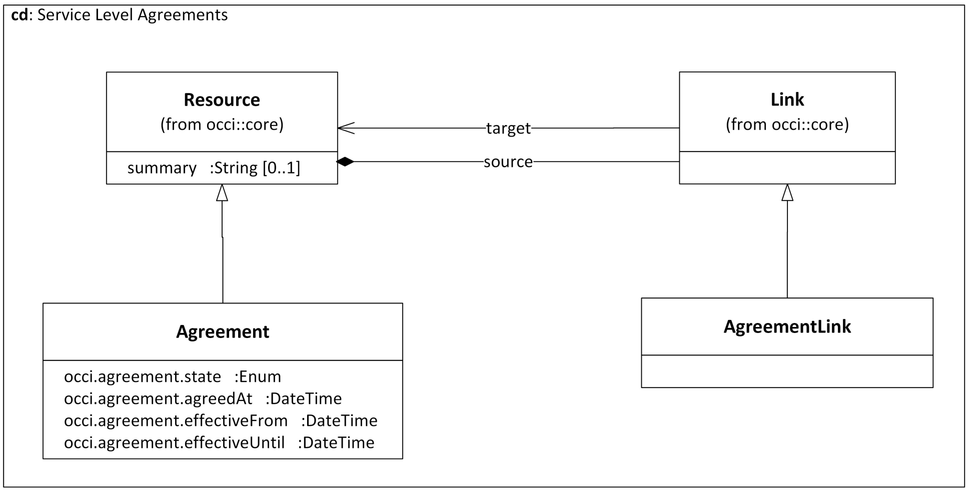
cd: Service Level Agreements
Resource
(from occi::core)
summary :String [0..1]
target
source
Link
(from occi::core)
Agreement
occi.agreement.state :Enum
occi.agreement.agreedAt :DateTime
occi.agreement.effectiveFrom :DateTime
occi.agreement.effectiveUntil :DateTime
AgreementLink

Figure 1. Overview diagram of OCCI Service Level Agreements types.

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.

It is REQUIRED by the OCCI Core Model specification that every type instantiated which is a sub-type of a Resource or a Link (i.e. Agreement and AgreementLink) MUST be assigned a Kind that identifies the instantiated type. To this end, each Kind instance MUST be related to the Resource or Link base type's Kind. That assigned Kind MUST be immutable to any client.

In the following table (Table 1) the Kind instances for the OCCI SLAs Resource, Link sub-types as well as the Mixins are introduced. For information on how to extend these types, please refer to the OCCI Core Model specification [2]. We also present related examples at the end of this document.

Table 1. The Kind instances defined for the SLAs 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	Related Kind
agreement	<schema>/sla#	A Service Level Agreement	<schema>/core#resource
agreement_link	<schema>/sla#	Link between a SLA and its associated resources	<schema>/core#link
agreement_tpl	<schema>/sla#	Mixin defining a SLA template collection	-
agreement_term	<schema>/sla#	Mixin defining a Term collection for an agreement	-

The following sections describe the Agreement and AgreementLink types, with details about their attributes, states and actions. The AgreementTemplate and AgreementTerm Mixins are also defined and presented. In the end, examples of OCCI SLAs instantiations are shown. These present several phases of the Service Level Agreement lifecycle, as well as specific instances of terms and service qualities.

3.1 Agreement

The Agreement type represents a generic contract resource which holds the information related to a SLA between a cloud service consumer and a provider for the provisioned resources (e.g. compute, storage, network etc.). The Agreement type inherits the Resource base-type defined in the OCCI Core Model [2]. The Kind instance assigned to the Agreement type is *http://schemas.ogf.org/occi/sla#agreement*. An Agreement instance MUST relate and expose this Kind.

Table 2 describes the attributes defined by the Agreement type through its Kind instance. These attributes MUST be exposed by an instance of the Agreement type. In Figure 2 the allowed states of an Agreement instance are presented. Those specific states MUST be assigned to an Agreement instance by a cloud service provider SHOULD the implements the OCCI SLAs specification.

Table 2. Attributes defined for the Agreement type.

Attribute	Type	Multiplicity	Mutability	Description
occi.agreement.state	Enum {Pending, Accepted, Rejected, Suspended, Terminated}	1	Immutable	Current state of the instance.
occi.agreement.agreedAt	Datetime (ISO8601)	0. . . 1	Immutable	The point in time when the agreement was made.
occi.agreement.effectiveFrom	Datetime (ISO8601)	0. . . 1	Mutable	The point in time when the agreement's effectiveness begins.
occi.agreement.effectiveUntil	Datetime (ISO8601)	0. . . 1	Mutable	The point in time when the agreement's effectiveness ends.

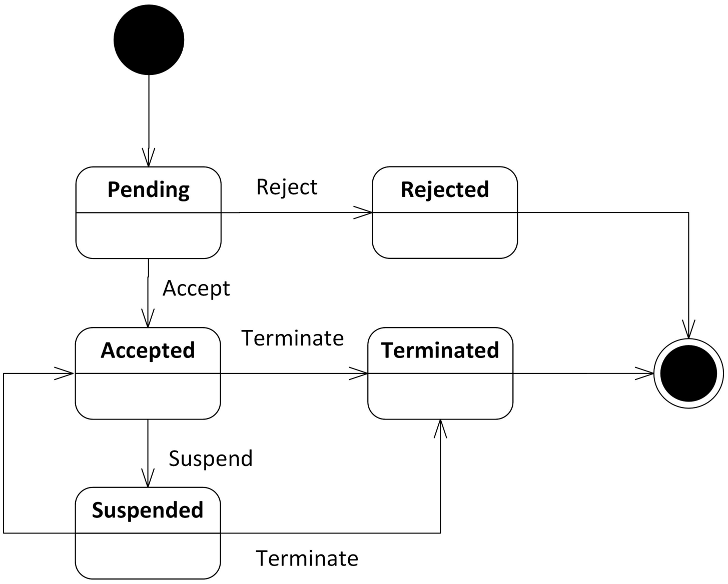

Figure 2. State diagram for Agreement instance, inspired by WS-Agreement states [4] .

The actions that are applicable to Agreement instances are presented in Table 3. The Actions are defined by the Kind instance *http://schemas.ogf.org/occi/sla#agreement*. Every Action in the table is identified by a Category instance using the *http://schemas.ogf.org/occi/sla#* categorization scheme. The "Action Term" below refers to the term of the Action's Category identifier.

Table 3. Actions applicable to instances of the Agreement type.

Action Term	Target state	Attributes
accept	Accepted	–
reject	Rejected	–
suspend	Suspended	–
unsuspend	Accepted	–
terminate	Terminated	–

These actions MUST be exposed by an instance of Agreement type of an OCCI SLAs implementation. The implementation of the Agreement type is REQUIRED if a cloud service provider adopts the OCCI SLAs specification.

3.1.1 AgreementTemplate Mixin

In order to allow the classification of agreements and the provisioning of Service Level Agreement templates, an OCCI Mixin is introduced. The AgreementTemplate Mixin is assigned the "scheme" *http://schemas.ogf.org/occi/sla#* and the term agreement_tpl. An AgreementTemplate mixin MUST support these values. The use and instantiation of this Mixin is OPTIONAL but RECOMMENDED for improved classification and management of the agreements. There are no specific attributes defined for the AgreementTemplate Mixin, thus every provider that implements the OCCI SLAs specification MAY introduce provider specific attributes using the Attributes Set inherited from the Category type.

As can be seen in the example diagram bellow, the AgreementTemplate mixin can be used either for simple agreement tagging (e.g. gold, silver etc.) of a Collection but also for introducing specific attributes and features for each tag.

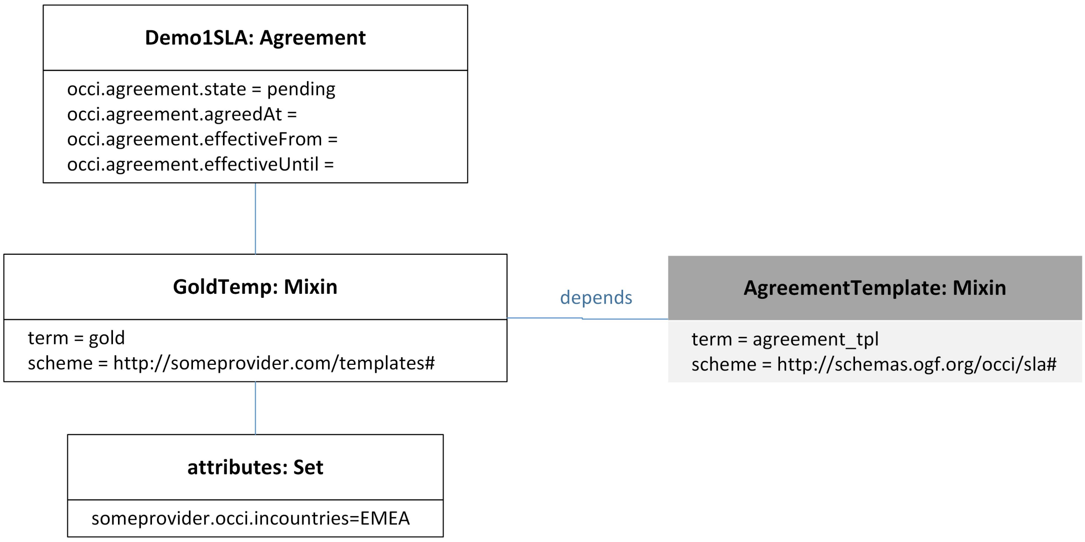
Demo1SLA: Agreement
occi.agreement.state = pending
occi.agreement.agreedAt =
occi.agreement.effectiveFrom =
occi.agreement.effectiveUntil =
GoldTemp: Mixin
term = gold
scheme = http://someprovider.com/templates#
depends
AgreementTemplate: Mixin
term = agreement_tpl
scheme = http://schemas.ogf.org/occi/sla#
attributes: Set
someprovider.occi.incountries=EMEA

Figure 3. Object diagram of an Agreement instance and its associated AgreementTemplate mixin.

3.1.2 AgreementTerm Mixin

A necessary part of an agreement offer, as well as the consequent agreement, is the section of the agreement term. To this end, the OCCI SLAs suggests the introduction of the agreement terms through the Mixin mechanism. The AgreementTerm Mixin is assigned the "scheme" *http://schemas.ogf.org/occi/sla#* and the term agreement_term. An AgreementTerm mixin MUST support these values. OCCI SLAs implementations SHOULD support this in order to provide a classification and definition mechanism for the various terms and conditions of the agreements. Therefore, the implementation of this functionality is OPTIONAL but RECOMMENDED.

Table 4 shows the defined attributes for the AgreementTerm Mixin. Following the rationale presented in the WS-Agreement specification [4] , OCCI SLAs defines two types of agreement terms: service terms and service level objectives (SLOs). The first includes information related with the service description and definition. The second refers to the guarantee terms that specify the service level which the two parties are agreeing to. A cloud service provider MAY introduce domain specific attributes to the AgreementTerm mixin instances that he constructs, through the attributes set inherited from the Category type. Mixin relationships MAY be used in order to enforce classification of capabilities but also to allow resource specific instantiation of AgreementTerm. For example, an availability Mixin could be defined, which is depended on the AgreementTerm Mixin type. The provider, then, MAY choose to instantiate different availability mixins for compute or storage resources (or any other offered resource) based on his own definition of availability for those resources.

The AgreementTerm state can be either *undefined*, *fulfilled* or *violated* (Figure 4). The undefined state is the initial state of the term until an assessment is made. During runtime and while the service and SLA is

Table 4. Attributes for the AgreementTerm Mixin.

Attribute	Type	Multi-plicity	Mutability	Description
occi.agreement.term.type	Enum {SERVICE-TERM,SLO, n/a}	1	Mutable	The type of the term that is being defined.
occi.agreement.term.state	Enum {Undefined,Fulfilled Violated}	1	Immutable	The state of fulfillment of the specific term.
occi.agreement.term.desc	String	0. . . 1	Mutable	The description of the agreement term defined with this mixin.

being monitored the state MUST be fulfilled or violated. When multiple terms exist (e.g. provider specific terms) then if at least one term in an agreement has state violated, then the agreement is considered violated (*occi.agreement.term.state=violated*).

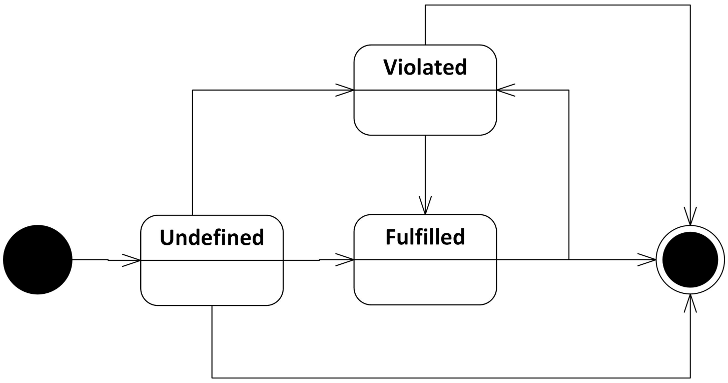

Figure 4. AgreementTerm state diagram.

In Figure 5 an example of using the AgreementTerm Mixin is shown. In the specific implementation an agreement offer (state: pending) is defined which describes a SLA for a compute service (memory: 16GB, cores: 4). The *Availability* Service Level Objective (SLO) is introduced through provider specific attributes in the respective mixin.

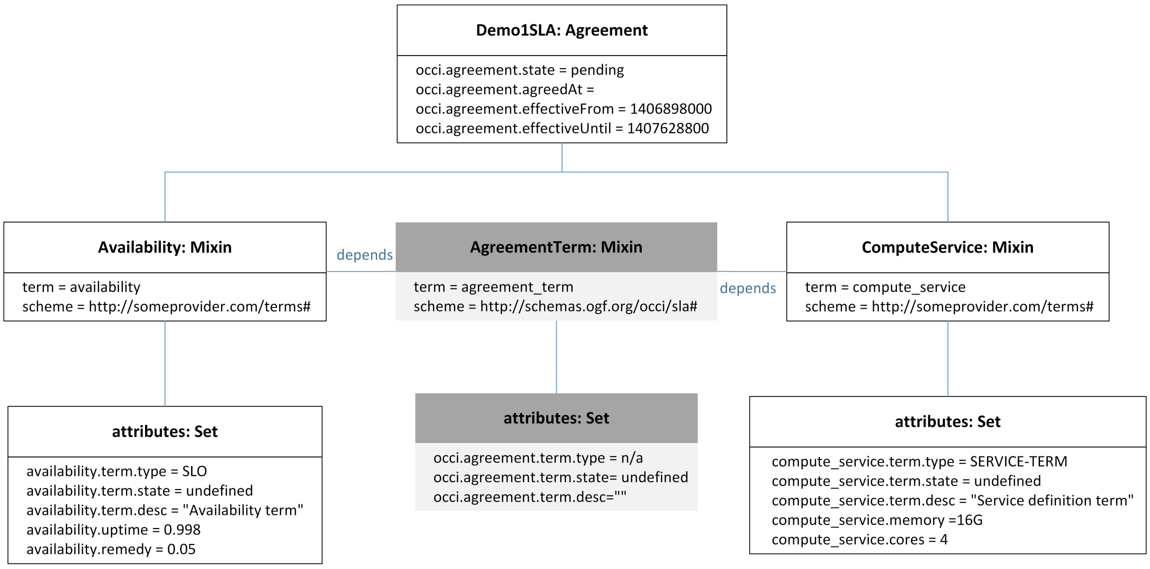
Demo1SLA: Agreement
occi.agreement.state = pending
occi.agreement.agreedAt =
occi.agreement.effectiveFrom = 1406898000
occi.agreement.effectiveUntil = 1407628800
Availability: Mixin
term = availability
scheme = http://someprovider.com/terms#
depends
AgreementTerm: Mixin
term = agreement_term
scheme = http://schemas.ogf.org/occi/sla#
depends
ComputeService: Mixin
term = compute_service
scheme = http://someprovider.com/terms#
attributes: Set
availability.term.type = SLO
availability.term.state = undefined
availability.term.desc = "Availability term"
availability.uptime = 0.998
availability.remedy = 0.05
attributes: Set
occi.agreement.term.type = n/a
occi.agreement.term.state= undefined
occi.agreement.term.desc=""
attributes: Set
compute_service.term.type = SERVICE-TERM
compute_service.term.state = undefined
compute_service.term.desc = "Service definition term"
compute_service.memory =16G
compute_service.cores = 4

Figure 5. Object diagram of an Agreement instance populated with AgreementTerm mixin.

3.2 AgreementLink

In order to associate signed Service Level Agreements with existing OCCI resource instances, the AgreementLink is introduced. This is a sub-type of the OCCI Core Model Link base type. Thus, the instantiation of an AgreementLink resource allows the linkage of resources of the previous defined Agreement sub-type with any OCCI Core Model Resource sub-type (e.g. Infrastructure sub-types). The implementation of the AgreementLink type is REQUIRED if a cloud service provider adopts the OCCI SLAs specification.

The AgreementLink type is assigned the Kind instance *http://schemas.ogf.org/occi/sla#agreement_link*. An AgreementLink instance MUST use and expose this Kind. The Kind instance assigned to the AgreementLink type MUST be related to the *http://schemas.ogf.org/occi/core#link* Kind.

Because of the multiple possibilities in terms of design and implementation of an OCCI compatible system, domain specific AgreementLink sub-types MAY be defined by cloud service providers. Thus, additional, provider specific attributes in such agreement link sub-types MAY be defined in by its Kinds instances.

3.3 OCCI Service Level Agreement example

In this section, an example instantiation of an Agreement type along with provider defined mixins is presented. It is to be noted that the implementation of an OCCI SLA framework is a responsibility of the cloud service provider. Thus, the instantiation of the proposed types and mixins are subject to the requirements and objectives of the provider. The presented instantiation of an OCCI SLA is only an example. Different approaches, mixins and attributes definitions could be followed.

The creation and provisioning of SLAs includes several phases. The process of reaching such agreement could be described by the following steps :

- Negotiation phase - The cloud service consumer retrieves the SLA templates, completes the REQUIRED values and submits an offer to the cloud service provider. (agreement-state: pending)
- Agreement phase - The cloud service provider can decide whether to accept the filled out template (the offer) or not. It is also possible to provide a counter-offer to the customer. (agreement-state: accepted, rejected, pending)
- Execution phase - When the agreement has been accepted the Agreement is in place and the (newly) created resource can be linked and associated with the reached agreement. (agreement-state: accepted)

The object diagram in Figure 6 represents an Agreement in the execution phase. In the presented example the Demo1SLA agreement is being populated with the SilverTemp mixin which is related to the AgreementTemplate Mixin type. This is used to tag and classify the agreement as well as to define some generic constraints such as the region in which the resources (under that SLA template) SHOULD be allocated. In addition to the template mixin several AgreementTerm mixins are defined either to define and describe the service offered or to introduce Service Level Objectives (SLOs) for the agreement.

To this end, through the *ComputeServiceTerm* mixin, the cloud service provider introduces a set of service terms which characterize the service being offered with this SLA. In this case it is a compute resource with technical specifications defined through provider-specific attributes (e.g. *compute_service.cores*, *compute_service.cpu* etc.). The *Availability*, *ServicePerformance* and *ServiceCapacity* are all Service Level Objective terms that set certain thresholds to metrics which determine the Quality of Service (QoS) of the respective offering. Every SLO term also defines the remedy value which is the compensation to the costumer in the event that the cloud service provider fails to meet the specified SLO. The value is usually a percentage of the agreed rate for the offered cloud service. The attributes defined in the mixins can be either mutable or immutable to the costumer depending on how the negotiation phase is being realized by the cloud service provider. What is more, every term has a current state value. Depending on the current assessment the terms are fulfilled or violated. Each violation will trigger the respective remedy value.

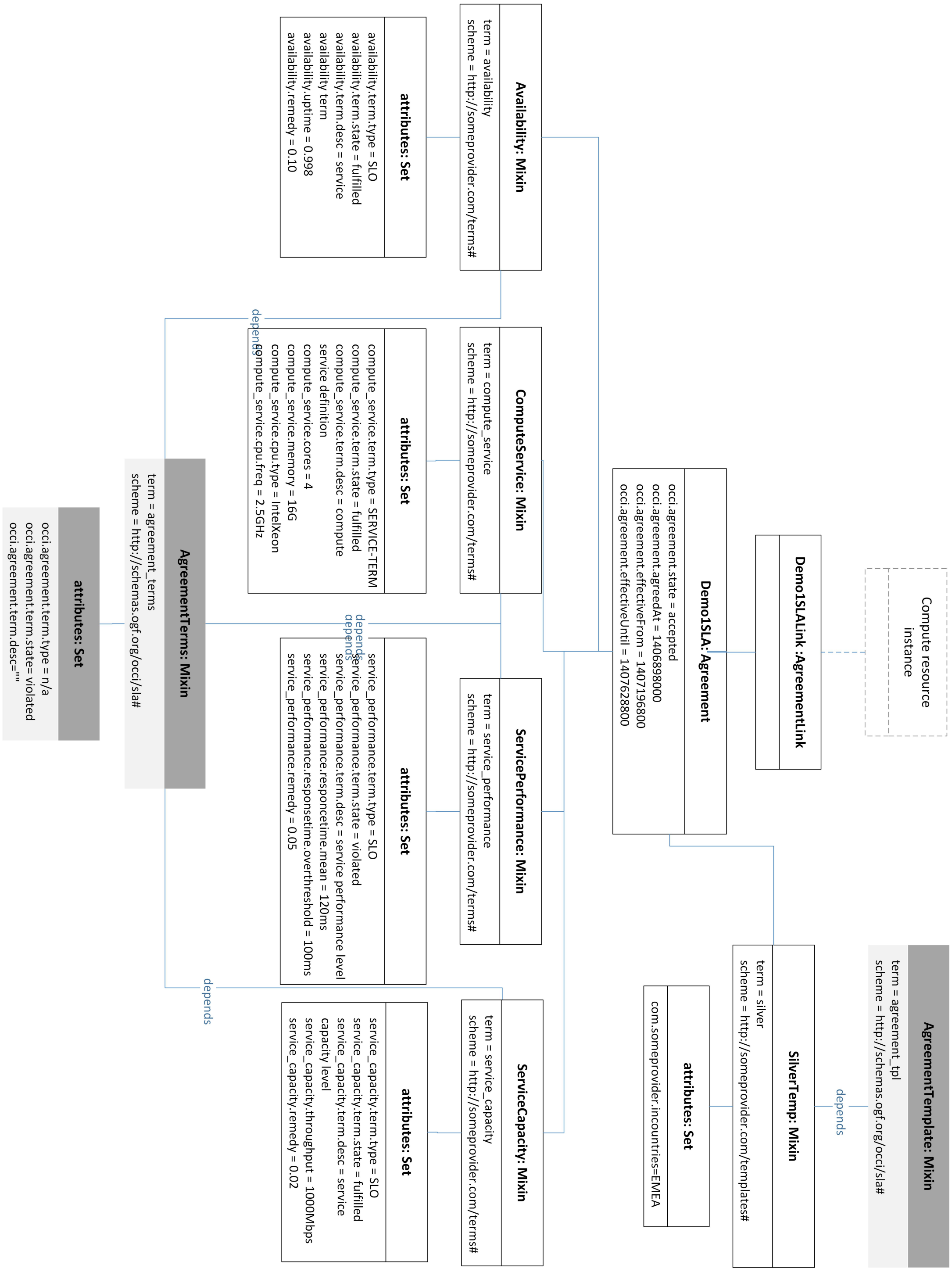
AgreementTemplate: Mixin
term = agreement_tpl
scheme = http://schemas.ogf.org/occi/sla#
depends
SilverTemp: Mixin
term = silver
scheme = http://someprovider.com/templates#
attributes: Set
com.someprovider.incountries=EMEA
Compute resource instance
Demo1SLALink :AgreementLink
Demo1SLA:Agreement
occi.agreement.state = accepted
occi.agreement.agreedAt = 1406898000
occi.agreement.effectiveFrom = 1407196800
occi.agreement.effectiveUntil = 1407628800
Availability: Mixin
term = availability
scheme = http://someprovider.com/terms#
attributes: Set
availability.term.type = SLO
availability.term.state = fulfilled
availability.term.desc = service
availability term
availability.uptime = 0.998
availability.remedy = 0.10
ComputeService: Mixin
term = compute_service
scheme = http://someprovider.com/terms#
attributes: Set
compute_service.term.type = SERVICE-TERM
compute_service.term.state = fulfilled
compute_service.term.desc = compute
service definition
compute_service.cores = 4
compute_service.memory = 16G
compute_service.cpu.type = IntelXeon
compute_service.cpu.freq = 2.5GHz
ServicePerformance: Mixin
term = service_performance
scheme = http://someprovider.com/terms#
attributes: Set
service_performance.term.type = SLO
service_performance.term.state = violated
service_performance.term.desc = service performance level
service_performance.responcetime.mean = 120ms
service_performance.responsetime.overthreshold = 100ms
service_performance.remedy = 0.05
ServiceCapacity: Mixin
term = service_capacity
scheme = http://someprovider.com/terms#
attributes: Set
service_capacity.term.type = SLO
service_capacity.term.state = fulfilled
service_capacity.term.desc = service
capacity level
service_capacity.throughput = 1000Mbps
service_capacity.remedy = 0.02
AgreementTerms: Mixin
term = agreement_terms
scheme = http://schemas.ogf.org/occi/sla#
attributes: Set
occi.agreement.term.type = n/a
occi.agreement.term.state= violated
occi.agreement.term.desc=""
depends
depends
depends
depends

Figure 6. OCCI SLA instantiation example.

4 Security Considerations

The OCCI Infrastructure 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.

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 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 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.
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.
Cloud service provider	The entity who offers a resource/service.
Cloud service consumer	The party which is in business relationship with the cloud service provider for using a cloud service/resource.
SLA	Service Level Agreement: the contract or agreement that the two parties (provider, consumer) need to "sign". It includes all the information about the services and the terms they both agree upon.
SLO	Service Level Objective: the quality of service aspect of the agreement. Specifies a non-functional guarantee in the SLA.
SLA Template	It is a resource that classifies set of terms and qualities for a provisioned service.

6 Contributors

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

Name	Affiliation	Contact
Gregory Katsaros	Intel	gregory.katsaros at intel.com
Thijs Metsch	Intel	thijs.metsch at intel.com
John Kennedy	Intel	john.m.kennedy at intel.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] 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] 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

[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] A. A. et. al, “Web services agreement specification (ws-agreement),” GFD-P-R.107, 2007. [Online]. Available: https://www.ogf.org/documents/GFD.107.pdf