

5 **Open Cloud Computing Interface – Service Level Agreements**

6 Status of this Document

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

9 Copyright Notice

10 Copyright © Open Grid Forum (2009-2016). All Rights Reserved.

11 Trademarks

12 OCCI is a trademark of the Open Grid Forum.

13 Abstract

14 This document, part of a document series produced by the OCCI working group within the Open Grid Forum
15 (OGF), provides a high-level definition of a Protocol and API. The document is based upon previously gathered
16 requirements and focuses on the scope of important capabilities required to support modern service offerings.

17 This document, part of a document series, produced by the OCCI working group within the Open Grid Forum
18 (OGF), provides a high-level definition of a Protocol and API in relation with the Service Level Agreements
19 extension of the OCCI Core Model. The document is based upon previously gathered requirements and focuses
20 on the scope of important capabilities required to support modern service offerings.

21 **Contents**

22	1 Introduction	3
23	2 Notational Conventions	3
24	3 Service Level Agreement	4
25	3.1 Agreement	5
26	3.1.1 AgreementTemplate Mixin	6
27	3.1.2 AgreementTerm Mixin	6
28	3.2 AgreementLink	8
29	3.3 OCCI Service Level Agreement example	8
30	4 Security Considerations	9
31	5 Glossary	11
32	6 Contributors	11
33	7 Intellectual Property Statement	12
34	8 Disclaimer	12
35	9 Full Copyright Notice	12

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 IaaS¹ 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 complementary 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. OCCI interaction occurs through *renderings* (including associated behaviors) and is expandable 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

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].

¹Infrastructure as a Service

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.

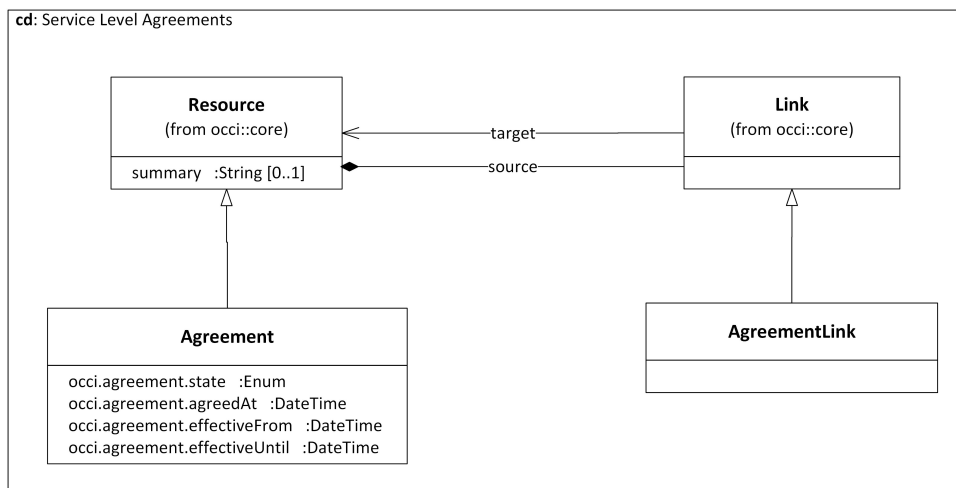


Figure 1. Overview diagram of OCCI Service Level Agreements types.

The **Agreement** type is a sub-type of the OCCI Core Model's **Resource** base type and inherits all its attributes. The HTTP Protocol [3] and Text Rendering [?] documents define 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.

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 2) 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.

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.

Table 2. 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	-

96 **3.1 Agreement**

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

102 Table 3 describes the attributes defined by the **Agreement** type through its **Kind** instance. These attributes
 103 **MUST** be exposed by an instance of the **Agreement** type. In Figure 2 the allowed states of an **Agreement**
 104 instance are presented. Those specific states **MUST** be assigned to an **Agreement** instance by a cloud service
 105 provider **SHOULD** the implements the OCCI SLAs specification. The **agreedAt**, **effectiveFrom** and **effectiveUntil**
 106 attributes **MUST** have an absolute datetime value (data, time or combined format) but **MUST NOT** represent
 107 a duration or time interval formatted value.

Table 3. Attributes defined for the **Agreement** type.

Attribute	Type	Multi- plicity	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.

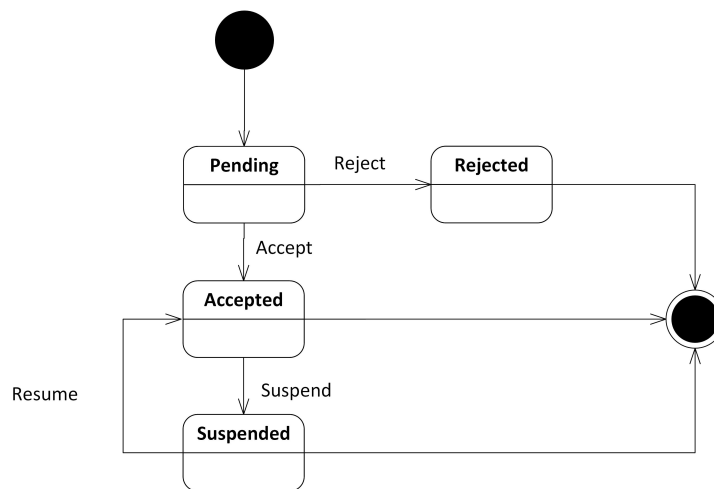


Figure 2. State diagram for Agreement instance, inspired by WS-Agreement states [4] .

108 The actions that are applicable to **Agreement** instances are presented in Table 4. The **Actions** are defined by

109 the **Kind** instance *http://schemas.ogf.org/occi/sla#agreement*. Every **Action** in the table is identified by a
 110 **Category** instance using the *http://schemas.ogf.org/occi/sla#* categorization scheme. The “Action Term”
 111 below refers to the term of the **Action**’s **Category** identifier.

Table 4. **Actions** applicable to instances of the **Agreement** type.

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

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

115 3.1.1 AgreementTemplate Mixin

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

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

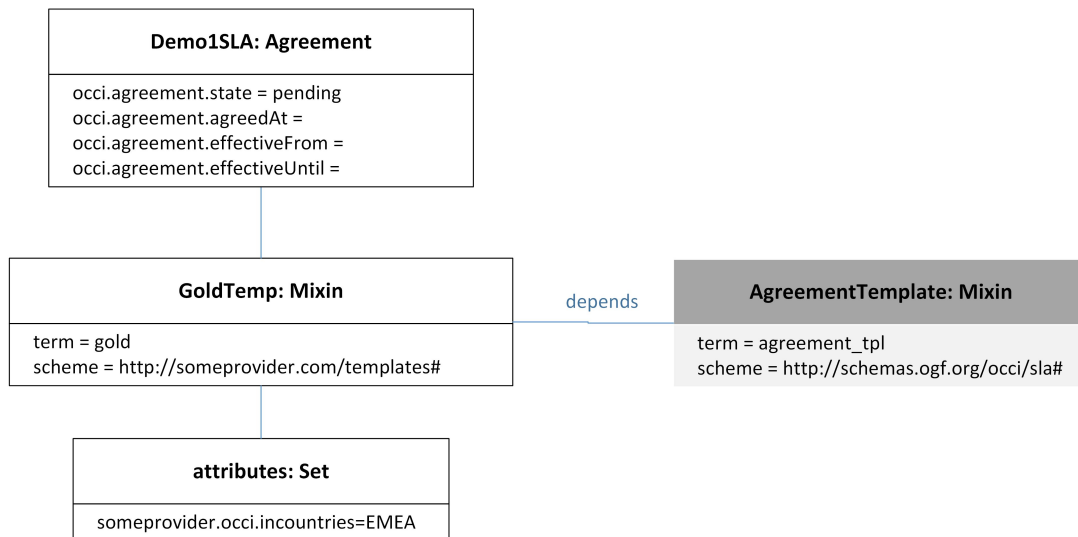


Figure 3. Object diagram of an Agreement instance and its associated AgreementTemplate mixin.

126 3.1.2 AgreementTerm Mixin

127 A necessary part of an agreement offer, as well as the consequent agreement, is the section of the agreement
 128 term. To this end, the OCCI SLAs introduces the agreement terms through the **Mixin** mechanism. The **Agree-**
 129 **mentTerm Mixin** is assigned the “scheme” *http://schemas.ogf.org/occi/sla#* and the term *agreement_term*.

130 An **AgreementTerm** mixin MUST support these values. OCCI SLAs implementations SHOULD support this
 131 in order to provide a classification and definition mechanism for the various terms and conditions of the
 132 agreements. Therefore, the implementation of this functionality is OPTIONAL but RECOMMENDED.

133 While the Agreement Term Mixin as defined does not include any generic attribute, a provider specific term
 134 (e.g., availability, compute service term etc.) SHOULD be depended from the OCCI SLAs **AgreementTerm**
 135 **Mixin** and introduce a set of attributes that characterize those terms. In Table 5 a list of attributes is presented
 136 that a provider MAY use for the definition of the custom terms mixins. Following the rationale presented in
 137 the WS-Agreement specification [4] , OCCI SLAs defines two types of agreement terms: service terms and
 138 service level objectives (SLOs). The first includes information related with the service description and definition.
 139 The second refers to the guarantee terms that specify the service level which the two parties are agreeing
 140 to. A cloud service provider MAY introduce more domain specific attributes to the **AgreementTerm** mixin
 141 instances that he constructs, through the attributes set inherited from the **Category** type. **Mixin** relationships
 142 MAY be used in order to enforce classification of capabilities but also to allow resource specific instantiation of
 143 **AgreementTerm**. For example, an availability **Mixin** could be defined, which is depended on the **AgreementTerm**
 144 **Mixin** type. The provider, then, MAY choose to instantiate different availability mixins for compute or storage
 145 resources (or any other offered resource) based on his own definition of availability for those resources.

Table 5. Suggested **Attributes** for a provider-defined **AgreementTerm** Mixin.

Attribute	Type	Multi- plicity	Mutability	Description
{term_name}.term.type	Enum {SERVICE-TERM,SLO-TERM, n/a}	1	Immutable	The type of the term that is being defined.
{term_name}.term.state	Enum {Undefined, Fulfilled, Violated}	1	Immutable	The state of fulfillment of the specific term.
{term_name}.term.desc	String	0...1	Immutable	The description of the agreement term defined with this mixin.
{term_name}.term.remedy	String	0...1	Immutable	The remedy value (e.g., price penalty) or action e.g., command) when an SLO term is being violated.

146 The **AgreementTerm** state can be either *undefined*, *fulfilled* or *violated* (Figure 4). The undefined state is
 147 the initial state of the term until an assessment is made. During runtime and while the service and SLA is
 148 being monitored the state MUST be fulfilled or violated. When multiple terms exist (e.g., provider specific
 149 terms) then if at least one term in an agreement has state violated, then the agreement is considered violated
 150 ({term_name}.term.state=violated).

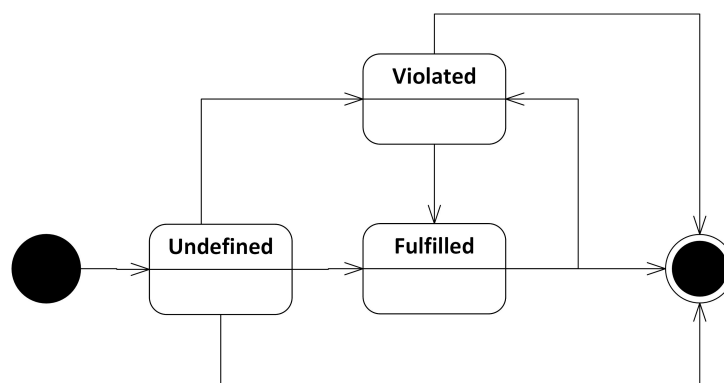


Figure 4. AgreementTerm state diagram.

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

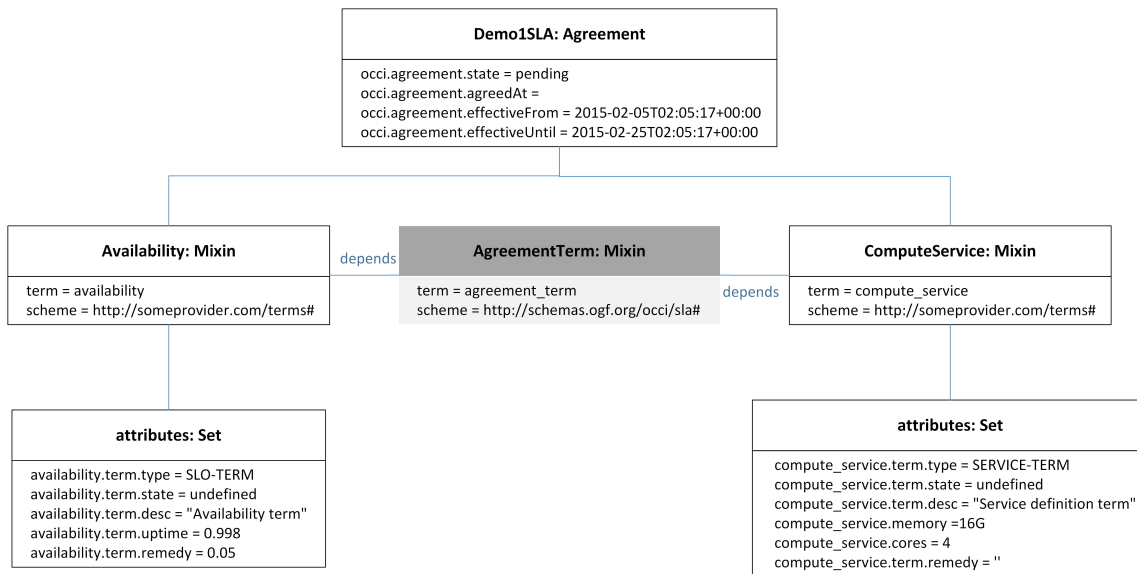


Figure 5. Object diagram of an Agreement instance populated with AgreementTerm mixin.

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)

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

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

199 4 Security Considerations

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

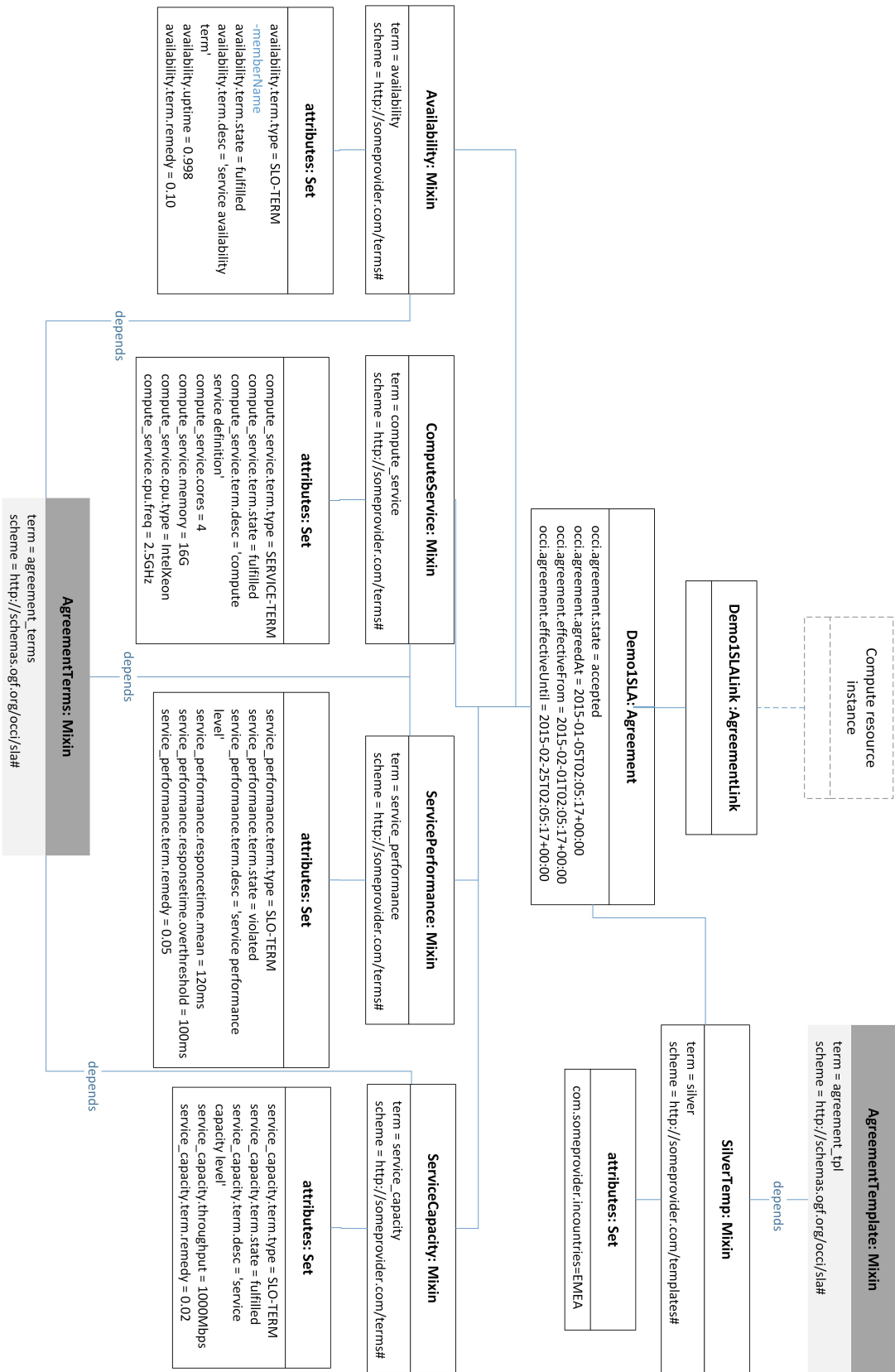


Figure 6. OCCI SLA instantiation example.

202 5 Glossary

Term	Description
Action	An OCCI base type. Represents an invocable operation on an 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 <i>entity instance</i> 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.
203 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 <i>entity instance</i> . The “mix-in” concept as used by OCCI <i>only</i> 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 <i>entity instance</i> . 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.
204 URN	Uniform Resource Name.

205 6 Contributors

206 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
207 John Kennedy	Intel	john.m.kennedy at intel.com
Alexander Stanik	TU Berlin	alexander.stanik at tu-berlin.de
Wolfgang Ziegler	SCAI Fraunhofer	wolfgang.ziegler at scai.fraunhofer.de

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

209 **7 Intellectual Property Statement**

210 The OGF takes no position regarding the validity or scope of any intellectual property or other rights that
211 might be claimed to pertain to the implementation or use of the technology described in this document or the
212 extent to which any license under such rights might or might not be available; neither does it represent that
213 it has made any effort to identify any such rights. Copies of claims of rights made available for publication
214 and any assurances of licenses to be made available, or the result of an attempt made to obtain a general
215 license or permission for the use of such proprietary rights by implementers or users of this specification can be
216 obtained from the OGF Secretariat.

217 The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications,
218 or other proprietary rights which may cover technology that may be required to practice this recommendation.
219 Please address the information to the OGF Executive Director.

220 **8 Disclaimer**

221 This document and the information contained herein is provided on an "As Is" basis and the OGF disclaims all
222 warranties, express or implied, including but not limited to any warranty that the use of the information herein
223 will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

224 **9 Full Copyright Notice**

225 Copyright © Open Grid Forum (2009-2016). All Rights Reserved.

226 This document and translations of it may be copied and furnished to others, and derivative works that comment
227 on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in
228 whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph
229 are included as references to the derived portions on all such copies and derivative works. The published OGF
230 document from which such works are derived, however, may not be modified in any way, such as by removing
231 the copyright notice or references to the OGF or other organizations, except as needed for the purpose of
232 developing new or updated OGF documents in conformance with the procedures defined in the OGF Document
233 Process, or as required to translate it into languages other than English. OGF, with the approval of its board,
234 may remove this restriction for inclusion of OGF document content for the purpose of producing standards in
235 cooperation with other international standards bodies.

236 The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or
237 assignees.

238 **References**

- 239 [1] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119 (Best Current Practice),
240 Internet Engineering Task Force, Mar. 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2119.txt>
- 241 [2] R. Nyrén, A. Edmonds, A. Papaspyrou, and T. Metsch, "Open Cloud Computing Interface – Core," Open
242 Grid Forum, September 2016. [Online]. Available: <https://www.ogf.org/documents/GFD.221.pdf>
- 243 [3] R. Nyren, T. Metsch, and A. Edmonds, "Open Cloud Computing Interface – HTTP Protocol," Open Grid
244 Forum, September 2016. [Online]. Available: <https://www.ogf.org/documents/GFD.223.pdf>
- 245 [4] A. A. et. al, "Web services agreement specification (ws-agreement)," GFD-P-R.107, 2007. [Online].
246 Available: <https://www.ogf.org/documents/GFD.107.pdf>