GWD-I
OCCI-WG
occi-wg@ogf.org

Thijs Metsch, Platform
Victor Bayon, Intel
Andy Edmonds, Intel
Alexander Papaspyrou, TUDO
April 2011

SLA Agreement, Negotiation, Execution and Monitoring using OCCI

Status of This Document

Group Working Draft (GWD)

Document Change History

1. October 2011 - Publication as a Google Doc as editable document
2. April 2012 - Draft as a LaTeX file posted to svn with inline comments and addenda from the Google Doc
3. July 2012 - Sanitized version

This template has been updated to conform to GFD-C.152 [**?**].

Copyright Notice

Trademark

Abstract

Informative document abstract of 1-2 paragraphs.

This document provides information to the Grid community [topic]. It addresses [main issues] and presents [outcome]. This document does not define any standards or technical recommendations.

Contents

Notice

This document is a working draft, obtained by juxtaposition of related comments: therefore it does contain inconsistencies that are neither indicated nor resolved. In the process of refining the paper they will converge to an optimal (and consistent) solution. Hopefully.

1 Introduction

This document describes how the Open Cloud Computing Interface (OCCI) can be extended to support mechanisms for Service Level Agreement and Monitoring. It is a first step towards creating extensions to the current version of the OCCI specification (Version 1.1) which enable OCCI-based Services to offer these features. This work is influenced by WS-Agreement & WS-Negotiation as well as several monitoring frameworks (references?).

This specification contains both details on SLAs and monitoring since these two aspects are closely related. It should be noted that — although related — the two specifications can exist independently. Ideally, where a system implements the OCCI notion of an SLA, it should also realize the exposure of a monitoring API with this specification's contained description of monitoring.

After describing the terminology, this document will introduce the negotiation of agreements, followed by the description of a Monitoring extension for OCCI.

2 Terminology

Throughout the document, we will use the following terms:

Service Provider ...

Costumer ...

Service Level Agreement (SLA) An agreement which defines a dynamically-established and dynamically managed automated contract/agreement between a service provider and a customer. The representation of this SLA is machine readable;

Service Level Objective (SLO) a technical clause that specifies a non-functional guarantee in legally bounding document (SLA), specifying the terms and conditions of service provisioning; a SLO is typically a collection of metrics related to the service, and of functions defined on such metrics.

Service level compliance test A function in a SLO evaluated as a boolean that represents the compliance to the agreed service level;

Service Level Monitoring (SLM) a technical clause that specifies the modality used to monitor service provisioning;

Agreement - A set of SLOs (reflected in a resource instances) and SLMs on which the client and service provider 'agree' (Part of an SLA).

Negotiation - The process of creating an agreement. This process can be partially- or fully-automated.

Template - An agreement template is a representation used by the service provider to advertise the types of SLOs a service provider is (possibly) willing to accept/provide. A template can be thought of as the service provider invitation to offers.

Offer - What the client requests to the provider, in terms of a sequence of SLOs instantiated using the template retrieved from the provider.

3 Negotiation and Provisioning of Agreements

The following subsections deal with negotiation and agreement and therefore also with the creation of the SLAs. The process of reaching an agreement, signing an SLA, and managing involved resources is described with this workflow:

Discovery phase The customer retrieves the Templates from one or more providers. The Template may consist of a "default" SLO that the provider is ready to accept, thus simplifying the following Negotiation phase. The costumer selects the templates fitting its request.

Negotiation phase The costumer instantiates the selected templates as SLOs representing the requested services (the offer). They are submitted to the Service Provider (SP), that may decide to accept the offer or not. It is also possible to provide a counter-offer to the customer, so that the phase iterate.

Execution phase When an agreement is reached, the resources (that may be created on purpose) are marked as falling under and associated with this agreement. Metrics and parameters included in the SLO are used to allocate the resource.

Monitoring(a) & Notification phase(b) Once resources are provisioned to the user, the SLM becomes effective. Upon violation, a Notification mechanism is used to notify the customer as indicated in the SLM. Monitoring (SLA compliance included) is considered as a part of the provided service, so that its features are included in the Template provided to the user.

A more detailed workflow is the following (NOT CONSISTENT WITH CURRENT CONTENT OF THE PAPER):

1. Retrieve a set of template mixins through the Query interface
2. Create a new resource instance of kind Agreement:
   (a) Request MUST include: Agreement Kind
   (b) MAY include Mixins from the templates (like GOLD)
   (c) MAY include additional/changed values to agree upon (Customization)
3. Service can respond:
   (a) with 200 OK and accept the agreement
   (b) redirect to a counter over
   (c) Bad Request and not accept the agreement
4. Client can now link to this agreement (Semenatics: add 1...* resource instances to an agreement)
   (a) Link must be of kind AgreementLink (rel is therefore: Agreement kind)
   (b) Link can be added to an existing resource instance, or to a new one
5. Service provider needs to take care that the resource instance linked to an agreement
   (a) have the right mixins so they can be monitored
   (b) monitor the resource instances an check against agreed values
   (c) report states (see state diagram)

3.1 Discovery phase

During the discovery phase the user queries the provider using a GET method, in order to obtain the description of available services.

```
HTTP GET /.well-known/org/ogf/occi/
```

The provider replies with a response (see table 1) containing a sample `agreement_template`, an entity whose role is to guide the generation of an acceptable offer from the user.

The `agreement_template` is a Mixin that represents a range of infrastructure Resources and Service Level Objectives provided to the user. It is an empty mixin, that needs to be associated with other Mixins, describing available resources and SLOs.

```
> GET /-/ HTTP/1.1
...
< Category: metric;
  scheme="http://iolanes.eu/occi/infrastructure#"[z];
  class="mixin";
  title="The metric mixin";
  attributes="timestamp{immutable} samplerate resolution unit"
< Category: rxtot;
  scheme="http://iolanes.eu/occi/infrastructure/metric/network#";
  class="mixin";
  title="net.rxkbtot";
  attributes="timestamp{immutable} samplerate resolution unit";
  rel="http://iolanes.eu/occi/infrastructure#metric";
  location="/metric/network/rxtot"
< Category: user;
  scheme="http://iolanes.eu/occi/infrastructure/metric/compute/cpu#";
  class="mixin";
  title="cpu.user";
  attributes="timestamp{immutable} samplerate resolution unit";
  rel="http://iolanes.eu/occi/infrastructure#metric";
  location="/metric/compute/cpu/user"
...
```

Table 1: Sample Query Interface Rendering of Two Metrics offered by the provider

The Mixins that are used to describe a Template SHOULD allow the introduction of ranges to indicate resource capabilities. This differentiates these mixins with respect to those used to describe allocated resources.

In addition to Resources templates, a provider MAY make available also SLO templates: such templates describe the available options for non-functional capabilities of Infrastructure Resources, and the monitoring capabilities associated with those functional capabilities.

```
> GET /-/
< Category: gold;
  scheme="http://";
  class="mixin";
  attributes="occi.compute.memory{value=1, type='int', immutable, required}"
```

It is the responsibility of the provider to allow the specification of all relevant and desirable aspects of a resource: e.g., a computing resource template should allow the specification of

Term	agreement_template
Scheme	`http://schemas.ogf.org/occi/sla#`
Title	OCCI SLA Template Mixin
Related	None
Actions	None
Attributes	None

Table 2: Definition of an `agreement_template`

a range of OSs.

The `agreement_template` is defined in table 2.

A simple example of a provider-specific SLA template for a computing resource is shown below:

Term	gold					
Scheme	`http://www.provider.com/infrastucture/templates#`					
Title	An example template					
Related	`http://schemas.ogf.org/occi/sla#agreement_template`					
Actions	N/A					
Attributes	[1] (j)	Attribute	Type	Multiplicity	Mutability	Description
		occi.Compute.cores	Integer	1	Immutable	Number of CPUs - default 8
		occi.compute.memory	float	1	Immutable	8.0

We consider that the pricing policy of the provider is to some extent present in the templates. The user is thus enabled to approximate the cost of a given provisioning. However, the exact cost of a given resource is established only at the end of the negotiation phase, before the user triggers resource allocation. Pricingdetails are Infrastructure Resource attributes, and are not considered in this paper.

3.2 Negotiation Phase

Based on the templates received during the Discovery Phase, the user decides whether it is appropriate to start a Service Level Agreement session with the provider. If so, the user sends a GET request to the provider in order to obtain an Agreement entity.

During such interaction, security is enforced throughout the rest of the session in order to enforce the validity of the SLA contract. The user is authenticated and identified. The **Location** field in the header is used in the response to indicate a URL for the agreement.

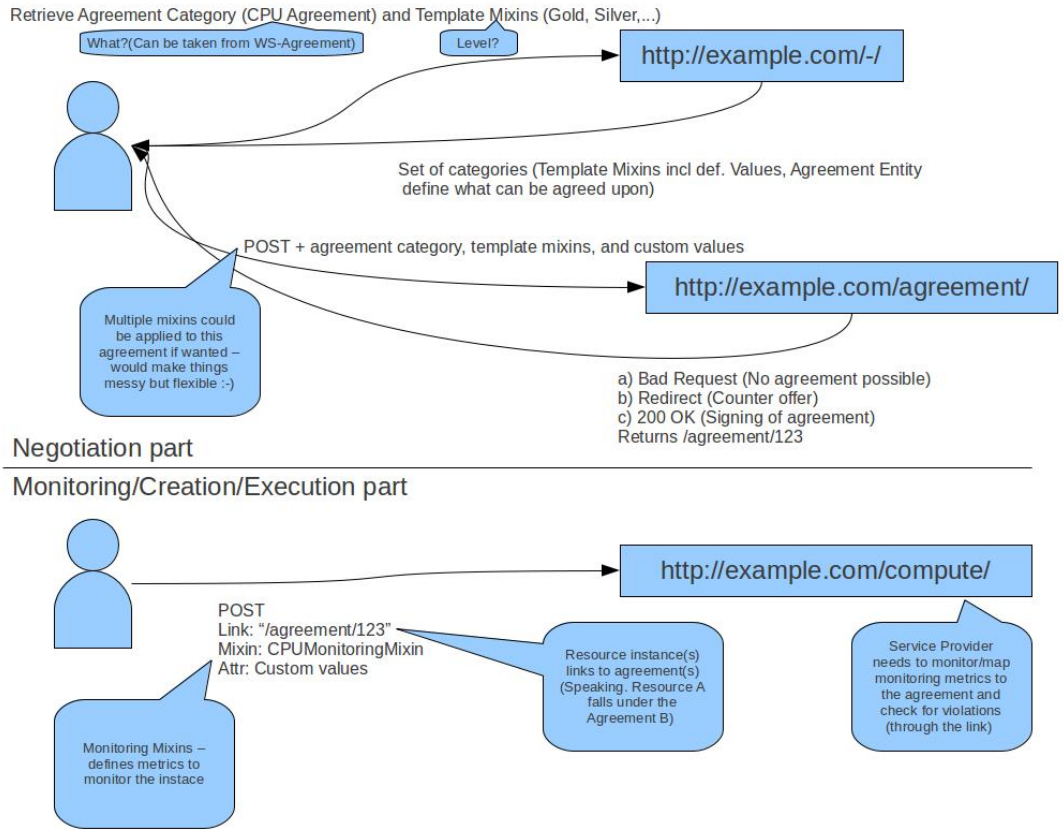

Figure 1: The negotiation process

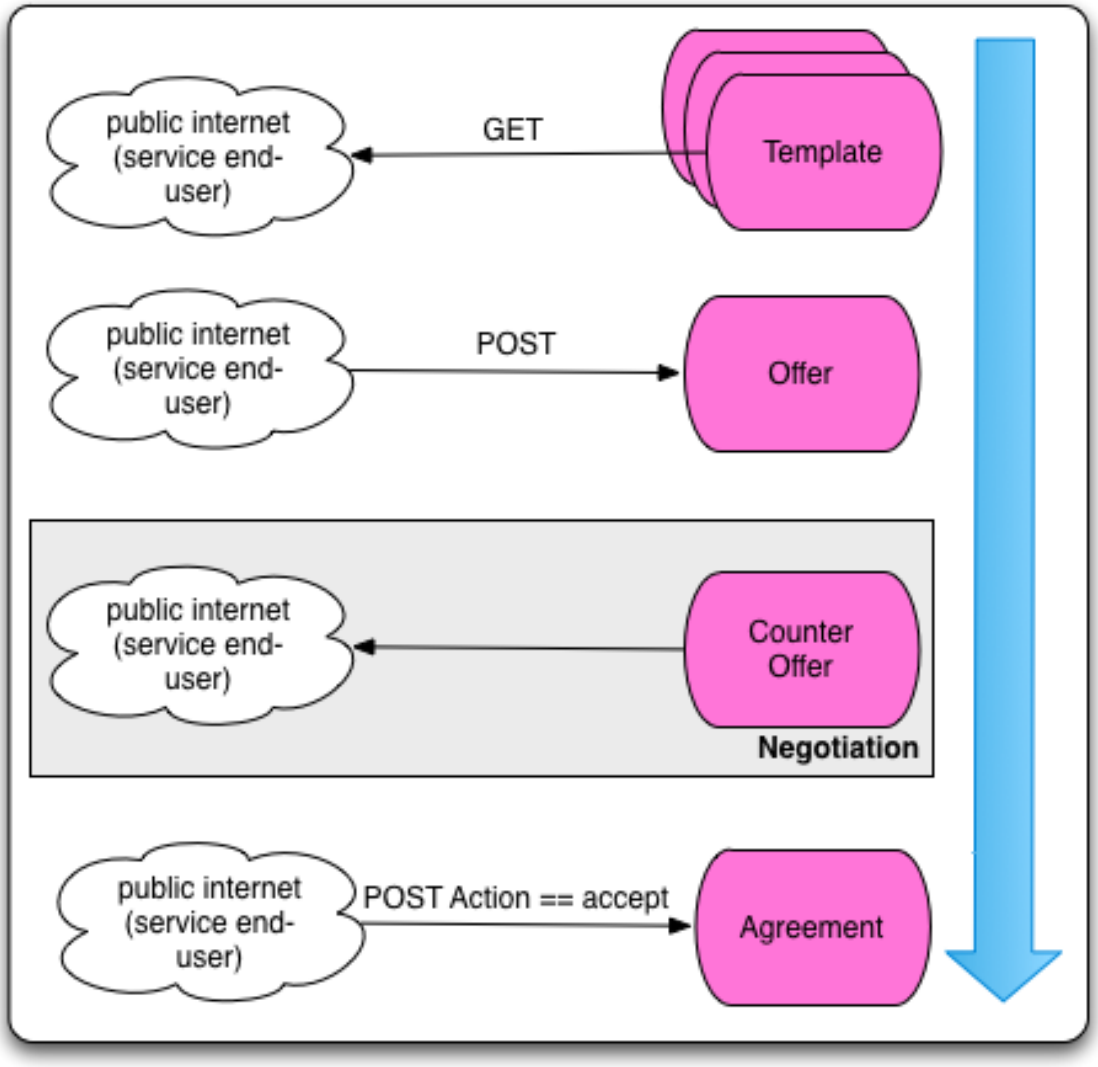

Figure 2: SLA agreement (to be updated)

Term	agreement				
Scheme	`http://schemas.ogf.org/occi/sla#`				
Title	An agreement				
Related	`http://schemas.ogf.org/occi/core#resource`				
Actions (c)	discover, sendOffer, negotiateOffer, acceptOffer, expire, fail				
Attributes	Attribute	Type	Multiplicity	Mutability	Description
	state	enum	1	Immutable	The state the agreement is in
	hash	string	1	Immutable	A hash to verification reasons filled after agree

Table 3: Definition of the `agreement` kind

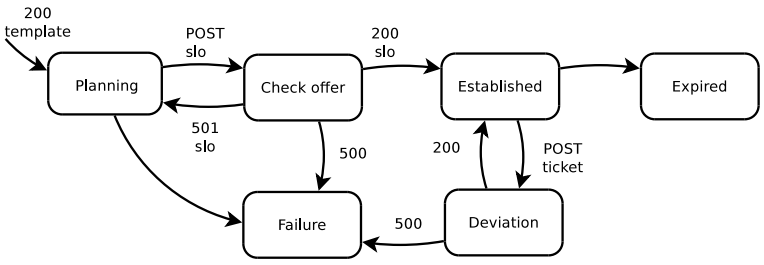

Figure 3: State diagram of the negotiation of a single SLO: each transition labeled with the HTTP verb (client request) or a status code (server response). Below the entity contained in the HTTP message.

At any point in time the user can issue a DELETE request to break the negotiation and remove agreement records.

Other relevant parameters of the agreement (like duration, form of payment etc.) are established.

The Agreement type (see the schema in table 3) inherits from the Resource type as defined in the OCCI Core Model.

At this point the user starts negotiating Resources and Service Level Objectives with the provider: the state diagram is represented in figure 3.

The user splits the agreement in a number of distinguished subsessions, each related with a single SLO. A SLO makes reference to several resources, but each resource is related to a single SLO (see figure 5. For each SLO, a distinguished instance of the diagram in figure 3 is run.

The user sends a POST requests to the provider, containing the **offer** for the specific SLO

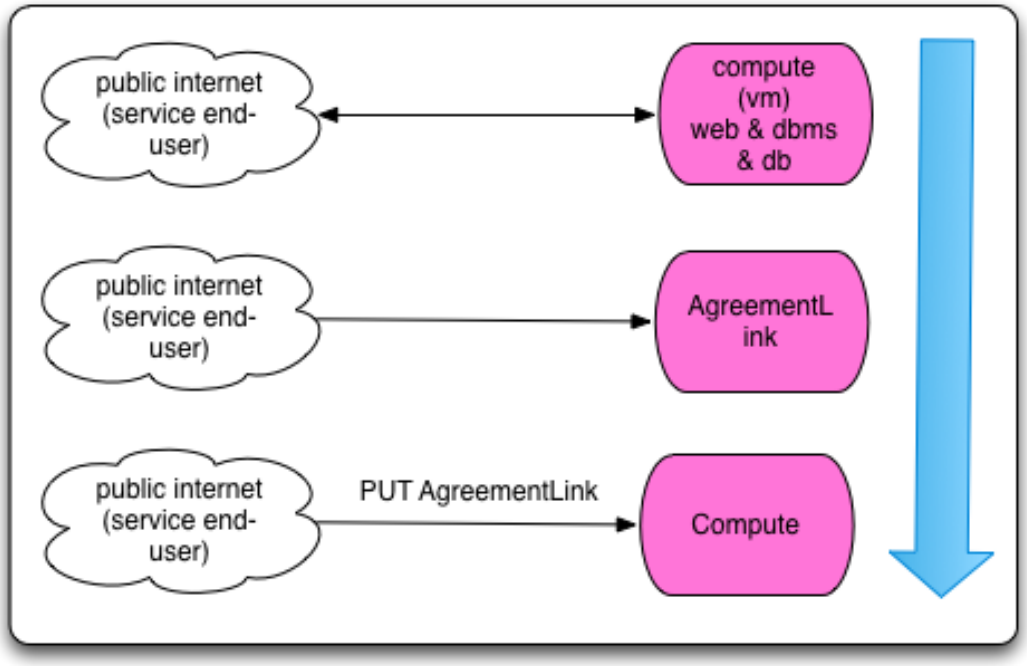

Figure 4: SLA provisioning

and for the related resources. The offer is subordinated (according with POST operation) to the URL representing the Agreement session.

1. Existing or newly created resources can now be bound to the agreement using the AgreeementLink.
2. `HTTP POST /agreement/` (with Category of the resource and a link description)
3. `HTTP POST /agreement/link/` (with Kind of the AgreementLink and source and target attributes)
4. `HTTP POST /agreement/123` (with the Link definition)

If the provider accepts the offer, an OK(200) response is returned, with SLOs and resources fully defined. Default values are introduced for relevant parameters left undefined in the offer. Here the Service provider needs to ensure that all criteria defined in the offer are met by the resource (e.g. Can it fall under the agreement, does it have the right monitoring mix-ins applied etc.)

The definition of resource entities is that described in the OCCI - Infrastructure document [**?**].

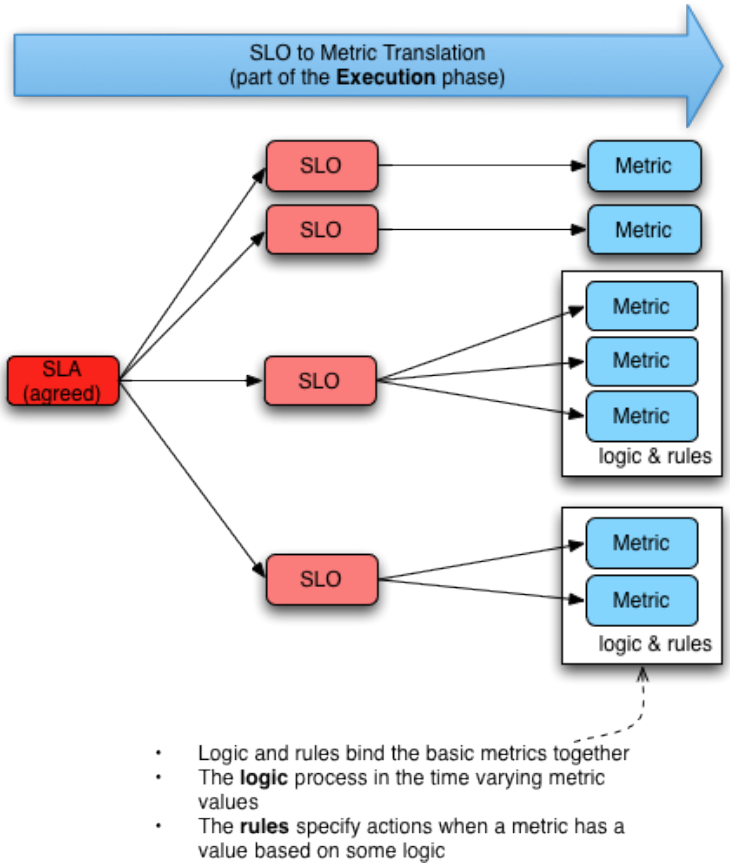

Figure 5: Relationships between SLA, SLO and metrics

Term	service_level_objective				
Scheme	`http://schemas.ogf.org/occi/sla#`				
Title	Non functional requirements and how to monitor them				
Related	`http://schemas.ogf.org/occi/core#link`				
Actions	N/A				
Attributes	Attribute	Type	Multiplicity	Mutability	Description

Table 4: Schema of the Service Level Objective

A Service Level Objective is described as a Link between an Agreement and one or more resources (see the schema in table **??**. This link will associate resource instances with the agreement.

The SLO is defined by mixins: among them, there are those that specify resource reservation details, metrics that define the service level, and monitoring details. It is the responsibility of the provider to introduce as mixins of the SLO all relevant parameters.

In figure 3 we describe SLO management: each state represents local actions on client or provider side or both, while state transitions involve communication. The following states are defined for each SLO operation instance:

- planning - the client has received the template, and is elaborating an offer;
- check offer - the provider is checking the offer received by the client against available resources; in case the provider cannot meet the offer, the system may rollback to the planning state;
- established - the resources have been granted to the client that starts using them;
- expired - the SLO meets the conditions agreed for its expiration.
- processing deviation - the provided service measurably deviates from the SLO. The deviation is analyzed and compensating actions are undertaken, possibly closing the session.

When all of the SLOs are successfully instantiated, the user declares that the agreement is in action by triggering the Agree action of the Agreement resource, thus steppping into the execution phase.

At any time the user can issue a DELETE for a specific SLO, with the effect of removing also the resources associated with the SLO itself: therefore the SLO and its associated resources share the same lifecycle. To modify an implemented SLO the user can modify mutable attributes in the SLO or in the resource.

Term	metric				
Scheme	`http://schemas.ogf.org/occi/monitoring#`				
Title	A metric mixin				
Related	None				
Actions					
Attributes	Attribute	Type	Multiplicity	Mutability	Description
	value	string	1	Immutable	Based on unit, i value's type.
	timestamp	string[n]	1	Immutable	ISO8601
	timestamp.msec	float	1	Immutable	msecs past timestan
	sampleperiod	float	0..1	Mutable	seconds
	resolution	string	0..1	Mutable	SI prefix
	unit	string	1	Immutable	If the metric bein tored is an OCCI a (e.g. occi.compute.r then the designat is used. Otherw provider defined app unit.
	sensor	string	Immutable	sensor type	
	sensor.parms	string	Immutable	parameters for the sensor	

Table 5: A sample monitoring metric mixin

In next sections further details of SLO mixins, and a simple use case.

3.2.1 The metric mixins

A Metric Mixin is used to complement a resource with a metric used for Service Level Agreement. The metric Mixin is added to the Service Level Objective linking to the resource the metric refers to. The attributes of a Metric describe aspects like measurement unit, or the application/method used to perform the measurement. Certain methods may need the specification of a series of parameters with a given syntax. These details are dealt with in specific "monitoring method" profile documents. A Metric corresponds to a single value. The schema of a sample monitoring mixin is in table 6.

3.2.2 The monitoring mixins

The Monitoring mixin describes the algebraic and logic functions applied to measurements to provide the monitoring service to the user, that here we call *calculated metrics*: the presence

of at least one calculated metric returning a boolean value (the *Rule*) is mandatory for a Monitoring Mixin.

Calculated metrics are metrics associated with a Resource that are evaluated starting from raw metrics. A calculated metric is the output of a Processor Resource. A Processor Resource represents the entity (e.g. software entity) that applies a function on 1 or more inputs. A Processor Resource has one or more raw inputs (Defined by mixin metrics). A Processor Resource has one output. A Processor Resource is linked to a Metric. When the system (implementation) reads a Processor it returns the calculated value. This allows the calculated value be offered as a Metric associated with a monitored Resource. If the Metric is a calculated metric then the mixin definition of that Metric will have its rel value pointing to the Processor Resource type (kind) e.g.

```
category: occi.core.cpu.cost;
scheme="http://schemas.ogf.org/occi/metric#";
class="mixin";
rel="http://schemas.ogf.org/occi/processor#cpu_billing_calculator"
```

Only a single rel value is needed if processors can be represented as a graph of collaborating processors with the sole purpose of outputting a single value which is used as the "raw" metric value. This would follow a similar approach taken with WS-BPEL where the internal interfaces that make up the composition are of no concern to the client but only the "terminal" external interface.

A Monitoring mixin defines also whether data are pushed to the user as notifications, or whether data is made available upon request: this aspect is discussed in section 3.4.

A Monitoring mixin MUST contain an boolean attribute `Active`, and two actions `Activate` and `Deactivate`. When monitoring is deactivated the user will not exert service level monitoring, and the monitoring infrastructure (sensors and monitoring agents) will not be allocated for that user.

3.2.3 The audit mixins

This is a general purpose mixin that allows date stamping of certain operations related to Agreement or SLO establishment.

Term	monitoring
Scheme	`http://schemas.ogf.org/occi/monitoring#`
Title	A monitoring mixin
Related	None
Actions	
Attributes	

Attribute	Type	Multiplicity	Mutability	Description
active	boolean	1	Mutable	
calculator	string	1	Immutable	how the monitoring value is computed
delivery_profile	string	1	mutable	specifies how monitoring is delivered (better done with a mixin)
delivery_parms	string	1	Mutable	parameters for the specific monitoring profile

Table 6: A sample monitoring mixin

Term	date_audit				
Scheme	`http://schemas.ogf.org/occi/audit#`				
Title	Adds the date-time stamping to an associated instance				
Related	None				
Actions	None				
Attributes	Attribute	Type	Multiplicity	Mutability	Description
	end	string	0..1	Mutable	ISO8601
	end.msec	int	0..1	Mutable	
	created	string	1	Immutable	ISO8601
	created.msec	int	1	Imutable	
	updated	string	0..1	Immutable	ISO8601
	created.msec	int	0..1	Imutable	

3.2.4 A use case: a storage request

The case of the request of a storage service that is implemented with a single SLO.

1. The user asks the service provider for a SLO template for a storage resource;
2. The service provider delivers a template for the storage resources (e.g. size by enumeration) and for the SLO (e.g., seek time metric, updated EWMA every minute or notified by e.mail upon excess of a threshold);
3. The user fills the Template with values, thus specifying the SLO: the user specifies the requested resource (e.g. 10 GB) and a metric mixin for the SLO (for instance, seek time $<$ 5 msec) together with a monitoring mixin (e.g. e.mail when seek time $>$ 5 msec);
4. The provider policy (that cannot be specified in the template) is to deliver fast disks only above 100GB. So the provider returns a nack and a new filled template with 100Gb and 5ms; next another with 10Gb and 10ms, waitin for a positive ack from the user. When the alternatives are considered over, it terminates with a failure;
5. If the user receives a positive ack from the provider, it considers that the resource is available, and it will trigger an "agree" action on the agreement. The resources will be allocated and made accessible (using a link provided in the response) under this agreement. If it receives a negative ack followed by a counter offer, it tries to decide whether the counter offer is acceptable or not. If acceptable, the SLO description is returned to the provider. After a timeout the user terminates the session with a failure.
6. When the provider succeeds in allocating the resource, it also activates the sensors, that will perform the seek time measurement and report the user in case of problems.

3.3 Execution Phase

... empty ...

3.4 Monitoring and Notification Phase

Once a service consumer has running instances with a provider, the next thing that this consumer should be able to do is monitoring those instances to proactively manage potential problems. As such—from a point of view of manageability—monitoring is essential. The approach taken in this monitoring extension is to keep things as simple as possible and stay compatible with the Core model in that the defined Mixin can be validly applied to all instances of Entity.

A provider that offers SLAs SHOULD monitor the delivered service instances with respect to the metrics used in the SLO (see figure 5). This is not needed in the case agreement is reached on a "best effort" basis, or when the client is not willing to monitor the provided service, since monitoring may have an impact on the cost. However, when a SLA that integrates monitoring enters the established state, the provider MUST setup the required monitoring so to produce the agreed metrics.

The provider MAY include in the SLO other metrics whose values are not bound by the SLA: this service is useful for the user that wants to optimize the utilization of the provided infrastructure for a specific application.

A SLO mixin MAY contain attributes that are defined as `Mutable`: the user can modify such attributes while the SLA is in action. For instance, this may be useful to disable certain notifications, upgrade a resource, tune a monitoring activity. The degree of flexibility MUST be agreed with the provider during negotiation. The way this feature is implemented is with a `PUT` request on the mutable attribute:

```
> POST /$RESOURCE/123-123-123
```

The way metrics are returned to the user is included in the SLO as well: notification styles range from the availability of an historical record to the push of a notification upon a deviation. Independently from the notification style, the service provider SHOULD make the indicated metrics accessible by the user. The consumer SHOULD have access to these SLOs as metrics using the OCCI monitoring specification.

The provider MUST give access to the last measurement through an attribute of the SLO that can be retrieved with a `GET`, unless the specific monitoring is not `Active`.

```
> GET /service/123-123?attribute=occi.service.messages.throughput[ae]
< 204 OK HTTP/1.1
```

Term	deviation_ticket				
Scheme	`http://schemas.ogf.org/occi/sla#`				
Title	Ticket describing the deviation of a resource from its SLO				
Related	`http://schemas.ogf.org/occi/core#resource`				
Actions	N/A				
Attributes	Attribute	Type	Multiplicity	Mutability	Description

Table 7: Schema of the `deviation_ticket`

```
...
< x-occi-attribute: occi.service.messages.throughput=2.0
```

alternative:

```
> GET /compute/123-123-123
> Category: occi.service.messages.throughput; scheme='http://iolanes.eu/infra/metrics#';
...
< x-occi-attribute: occi.service.messages.throughput = 2.0
```

However this modality is not sufficiently flexible. Other ways SHOULD be provided, according with specific monitoring profiles. The following list is not meant to be exhaustive:

- a dynamic web page (referenced in the SLO) is kept updated with metric and other data;
- an image or spreadsheet is kept updated with metric on a defined URL with a defined modality/structure;
- data can be accessed in a defined database;
- data are pushed to the user as a stream;
- defined events are sent to the user asynchronously.

Such modalities are discussed in specific profile documents: some of them allow to obtain an historical record of measurements on demand.

Upon detection of a SLA deviation, the user MAY file a *ticket* to the provider, in order to request a recovery action. The presence of a ticket is recorded to establish the business value of the provided service. In absence of an effective recovery action

Note that there is no direct relationship between the monitoring facilities deployed by the provider for internal use, and those deployed to meet user requests. Data representation may be distinct as well.

Resources

1. WSAG Specification: `http://ogf.org/documents/GFD.107.pdf`
2. Paper on restyfing WSAG: `http://www.ws-rest.org/2011/proc/a12-kubert.pdf`
3. A student's master thesis: Source N/A
4. SLA@SOI presentation: `http://dl.dropbox.com/u/165239/query-reservation.pptx`
5. WSAG Negotioation draft: `http://forge.gridforum.org/sf/docman/do/listDocuments/projects.graap-wg/docman.root.current_drafts.ws_agreement_negotiation_specifi`

A Appendix

A.1 Examples

A.1.1 Example Metric Mixins

Here a set of definitions for metrics Mixins would be cool (Maybe copy from WSAG-Negotiation)? Would be very useful to have WSAG mappings to OCCI mixins. In the current planned prototype the following metrics will be available:

how to have specify an instance's subresources in a category? suggestion:

```
< category: speed; scheme='http://iolanes.eu/infra/metrics#'; class='mixin'; location="/
```

Metric attribute defaults in QI sample rate, resolution, simple type, unit (rate, unit - from ganglia/rrdtool)

Metric history should be maintained - the duration is subject to SLA Deactivate all Resource Metrics

```
> DELETE /$RESOURCE/123-123-123
```

```
< category: occi.compute.*; scheme='http://iolanes.eu/infra/metrics#'; class='mixin'
```

Metric history should be maintained - the duration is subject to SLA

Retrieve a Ranged Metric Value Could we slightly abuse the content-range header? Specify a start and end timestamp as the value for the time range?

```
> GET /compute/123-123-123
> category: occi.compute.cpu.user; scheme='http://'; class='mixin'
> x-occi-attribute: occi.compute.cpu.user.rangefrom=897777337

> x-occi-attribute: occi.compute.cpu.user.rangeto=897788888



< ???
```

Inline

```
> GET /compute/123-123-123/metric/cpu/user?rangeFrom=XXX&rangeTo=XXX
```

Retrieve a Set of Metrics Multipart? CSV Retrieve a Metric Group A metric group is defined by a tag. If 2 or more metrics are associated with a tag then they're deemed to be a metric group. Two modes: X-OCCI-Location or inline data.

A.1.2 Compute Metrics

Compute Metrics Available in prototype:

- `cpu.user, cpu.sys, cpu.wait, cpu.lavg1, cpu.intsec, cpu.ctxsec,`
- `mem.tot, mem.buf, mem.used, mem.free, mem.cached, mem.swap`

These all MUST have their rel attribute set to `http://schemas.ogf.org/occi/monitoring#metric`

Title	Term	Scheme
cpu.user	user	`http://iolanes.eu/occi/infrastructure/metric/compute/cpu#`
cpu.sys	sys	`http://iolanes.eu/occi/infrastructure/metric/compute/cpu#`
cpu.wait	wait	`http://iolanes.eu/occi/infrastructure/metric/compute/cpu#`
cpu.lavg1	lavg1	`http://iolanes.eu/occi/infrastructure/metric/compute/cpu#`
cpu.intsec	intsec	`http://iolanes.eu/occi/infrastructure/metric/compute/cpu#[w]`
cpu.ctxsec	ctxsec	`http://iolanes.eu/occi/infrastructure/metric/compute/cpu#`
mem.tot	tot	`http://iolanes.eu/occi/infrastructure/metric/compute/memory#`
mem.buf	buf	`http://iolanes.eu/occi/infrastructure/metric/compute/memory#`
mem.used	used	`http://iolanes.eu/occi/infrastructure/metric/compute/memory#`
mem.free	free	`http://iolanes.eu/occi/infrastructure/metric/compute/memory#`
mem.cached	cached	`http://iolanes.eu/occi/infrastructure/metric/compute/memory#`
mem.swap	swap	`http://iolanes.eu/occi/infrastructure/metric/compute/memory#`

[w]Victor Bayon: intsec = Interrupts/Second ctxsec = ContextSwitch/Sec

A.1.3 Network Metrics(x)

Network Metrics Available in prototype: 1. net.rxkbtot, net.txkbtot

(x)Augusto Ciuffoletti:

The network metrics that are given in the draft are oriented to network interfaces, which can be considered as part of a computing device, and might be consistently indicated in this kind of mixin.

Instead network metrics should measure performance of networking resources, whatever they are. Typical metrics are point to point roundtrip times, gateway throughput, link bandwidth for an L2 device. Since one such type is defined in the "infrastructure" document, I'd better start with bandwidth, or a maximum RTT (which includes NICs characteristics), or packet loss rate etc.

Title	Term	Scheme
net.rxkbtot	rxtot	`http://iolanes.eu/occi/infrastructure/metric/network#`
net.txkbtot	txtot	`http://iolanes.eu/occi/infrastructure/metric/network#`

A.1.4 Storage Metrics

Storage Metrics Available in the prototype: 1. dsk.readtot, dsk.writetot, dsk.readkbtot,dsk.writekbtot

Title	Term	Scheme
dsk.readtot	readtot	`http://iolanes.eu/occi/infrastructure/metric/storage#`
dsk.writetot	writetot	`http://iolanes.eu/occi/infrastructure/metric/storage#`
dsk.readkbtot	readkbtot	`http://iolanes.eu/occi/infrastructure/metric/storage#`
dsk.writekbtot	writekbtot	`http://iolanes.eu/occi/infrastructure/metric/storage#`

A.2 Comparison with other similar initiatives

NEEDS TO BE CAREFULLY CHECKED (especially qupoted paragraphs)

OGF GRAAP-WG (ex WSAG)

The distinctive point is that OCCI-SLA strictly adheres to OCCI restful model, giving a uniform interface throughout the whole IaaS lifecycle.

The definition of agreement for WSAG-WG is:

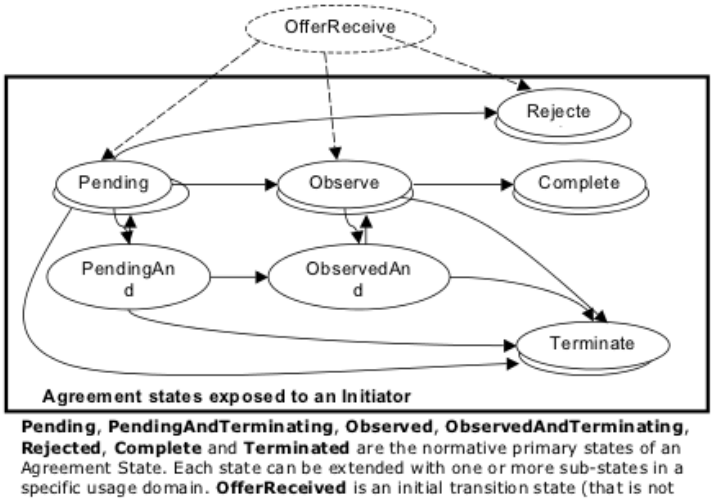

Figure 6: Agreement states exposed to an Initiator

> An agreement defines a dynamically-established and dynamically managed relationship between parties. The object of this relationship is the delivery of a service by one of the parties within the context of the agreement. The management of this delivery is achieved by agreeing on the respective roles, rights and obligations of the parties. The agreement may specify not only functional properties for identification or creation of the service, but also non-functional properties of the service such as performance or availability. Entities can dynamically establish and manage agreements via Web service interfaces.

Eh? WS agreement? why? :)

We should take a lokk at WSAG-Negotiation draft - WSAG spec itself has too much SOAP

Take the model!

States to be supported:

SLA@SOI SLA model

Link resource to agreement implies monitoring

monitoring a requirement

Example: verify that it is an 800MHz machine with an apache server with latency of 300ms

Latency of the webserver -¿ how are you going to check that?, latency from where? the client?

Applicable to hardware and software levels:

e.g. negotiate a VM running Apache

Provisioned resource is linked to agreement

WS-Agreement could be used as the content. OCCI attributes could be used within qualified by the
Extend mixins to allow for type and value info so that a resource template mixin can be used as the
Big bonus points for all this expressed as ABNF & ANTLR
With the current setup we could let fall several resource instances, of different kinds, fall under one a
What's the purpose of the SLA once agreed. Will it have rules or "smarts" to proactively resolve SL

SLASOI

Definition of agreement template:

> An agreement template is a (XML) document used by the agreement responder to advertise the types of offers it is willing to accept.

ITIL

Definition of agreement:

> An Agreement between a service provider and a customer. The SLA describes the service, documents service level targets, and specifies the responsibilities of the service provider and the customer. A single SLA may cover multiple services or multiple customers.

A.2.1 DMTF

todo

A.2.2 Others

From SLA Mangement Handbook:

> A Service Level Agreement (SLA) is an element of a formal, negotiated contract between two parties, viz., a Service Provider (SP) and a Customer. It documents the common understanding of all aspects of the ser- vice and the roles and responsibilities of both parties from service ordering to service termination. (source: SLA Management Handbook, TMForum) A single SLA may cover simple or compound services, or multiple cus- tomers. SLAs may be dynamic. Dynamic SLAs include one or more parameters that are variable. These vari- able parameters may be associated with agreed pricing scales.

A.3 Legacies...

Proposed changes to the OCCI specification [ak]Alexander Papaspyrou: In the process of making changes to HTTP, we could also do partial retrieval/update.

Andy Edmonds: and likely review content types/negotiation

- Alter the Query Interface so default values can be defined for templating
- add default value and include enums
- Add Notification mechanisms (look at SNIA's DDMI)
- Also to chunked transfer encoding

Required OCCI Extensions OCCI Mixin Extension 1. It is an extension to Category 2. Allow for type and value info so that a resource template mixin can be used as the resource template e.g. 3. Needed by SLA templates and resource template mixins to be complete

Category: gold; scheme="http://"; class="mixin"; attributes="occi.compute.memorydefaultvalue=1, type='int', immutable"

1. Also requirement to advertise enum values OCCI Filter Extension 1. Needed by monitoring. Allow for wild cards in the filter e.g.

```
GET /-/

> category: *; scheme="http://schemas.ogf.org/occi/infrastructure/metric#"; class="kind"
```

Open Issues

- How to ensure that resource instances have needed monitoring mixins assigned?
- What goes into kind agreement, agreement link, and actions, attribtues needed?
- What's with the history of an agreement? (aka. Agreement violated 2 days ago?)

Considerations

- other monitoring providers: amazon cloud watch, cloudkick, pingdom, See XDR in ganglia, however there is a problem with ganglia data definition. It defines a very flexible data format, but once is running everything is static, cannot change its configuration, ends up being too verbose.
- Units: if the metric being monitored is an OCCI attribute (e.g. occi.compute.memory) then the designated representational occi type (string, enum, integer, float, boolean)

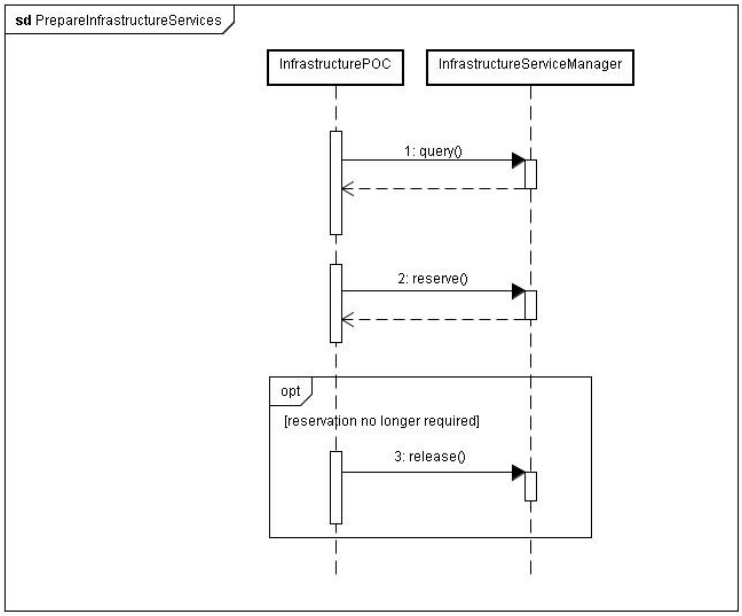

Figure 7: Message flow in a SLA.SOI document: we should make one similar...

is used. Otherwise, use provider defined appropriate representational occi type.

- It is likely better to link the metric to the attribute rather than duplicate this info
- Static versus dynamic (number of CPUs is not going to change rapidly per resource but CPU load yes) - sample rate defines this - it's fast (dynamic) or slow moving (static)
- metric resolution - magnitude (mega, giga, tera etc) and significant decimal places, readonly
- storage period (this could be also resampling rates per period of time, like RRDs)
- What about modification of these types?
- Depends on the client implementation
- Metric groups? could be done with OCCI tags (user mixin), not mandatory but capability exists already in core spec
- Rules and alerts - need to be related to SLA. Only have thresholds - breaking a threshold sends a message to another system for further processing [an](that when a threshold was exceeded, a user defined message could be sent). Rules are a can of worms, as there their definition could be very complex (e.g. time domain, event based, correlation, aggregation, etc). [an]Andy Edmonds: Look to use CDMI Queues here

- an import interface? not in the spec, for now a provider concern.
- In general, for metrics it may not be useful to offer a standardised set of metrics (given the large amount of possible metrics that could be offered) but rather we should specify a way that any metric fitting the metric model can be understood by a client.
- User defined metrics?

Attic I would love to see the following:

CostsMixin[ao]

[ao]Andy Edmonds: Is this better suited to a billing interface?

Andy Edmonds: Costs are calculated metrics. Currently the only consideration for metrics are "raw" metrics that are exposed by the resource and that there is a 1:1 mapping of resource attributes (e.g. occi.compute.memory) to metric

Thijs Metsch: If we define it that we - we should move it...what would be a reason for only allowing raw values? E.g. I'm thinking of MIPS, latency of a service - that'll be calculated too

Andy Edmonds: if you have calculated metrics the complexity of the current monitoring interface may increase. however that doesn't mean that the current interface as defined can't offer other metrics occi.compute.cpu.user, occi.compute.cpu.cost occi.compute.cpu.cost would be then linked to some aspect of a billing model/interface. in that sense, although it's calculated elsewhere, occi.compute.cpu.cost is raw

Thijs Metsch: the idea of defining the metrics not so much bound to the kinds of resource instance(s) is the reusability - but maybe this only applies to this billing mixin, and should be moved to a billing extension :-) (raw vs. calculated this is a point of discussion I guess).

Thijs Metsch: Also: what would be the costs/work to extend the complexity of the metrics interface to support this?

Thijs Metsch: costs stuff should be moved to billing ext.

This can now be supported as a calculated metric (derived)

Definition:

1. costs;`scheme=http://schemas.ogf.org/occi/billing`;

Attributes:

1. accumulate_costs (int)

2. costs_per_timeframe (int)
3. timeframe (long) in sec
4. currency (enum [$, EUR)]

Actions:

1. N/A