slalom

LEGAL & OPEN MODEL TERMS FOR CLOUD SLA AND CONTRACTS

SLA specification and reference model - b

D3.3

Dissemination level: Public

Work Package	**WP3, Technical Track**
Due Date:	M12 (31/12/2015)
Submission Date:	05/01/2015
Version:	1.1
Status	Final for submission (Updated after final review)
Author(s):	Nikos Bakalos (ICCS), George Kousiouris (ICCS), Dimosthenis Kyriazis (ICCS), Andreas Menychtas (ICCS), Emmanuel Protonotarios (ICCS), Theodora Varvarigou (ICCS), Oliver Barreto (ATOS), Ana Juan (ATOS), Aimilia Bantouna (UPRC), Panagiotis Demestichas (UPRC), Andreas Georgakopoulos (UPRC), Teta Stamati (UPRC), Kostas Tsagkaris (UPRC), Panagiotis Vlacheas (UPRC)
Reviewer(s)	Daniel Field (ATOS), Panagiotis Demestichas (UPRC)

The SLALOM Project is co-funded by the European Commission through the H2020 Programme under Grant Agreement 644720

CONTENTS

Tables

Figures

1 Introduction

The current document is the second one in the series of three deliverables of the SLALOM project that aims at proposing a specification for cloud Service Level Agreements (SLAs). The proposed SLA specification refers to the core SLA document that incorporates metrics (as specific objectives or quality attributes), parameters, rules as well as potential dependencies between rules. A JSON schema of the proposed SLA specification is also included in the document (aiming to provide a machine-readable format of the SLALOM proposition) along with practical examples of the proposed approach.

Comparing to the previous (initial) report, this document highlights and provides a concrete SLA specification proposition addressing the following:

- *SLA specification*: Following the analysis and assessment (through concrete SLA examples) of the structure of an SLA that was performed and presented in the previous report, this report provides the proposed SLA specification in terms of "blocks" of information and the corresponding fields. The work was based on the evolving ISO 19086-2 standard in terms of blocks and definitions for different metrics, parameters and rules. SLALOM proposes the adoption of these blocks, while it also proposes changes with respect to naming of specific elements as well as the inclusion of additional blocks and components in the SLA specification.
- *SLA metric definition*: Development of a formula / function that allows any provider to specify any metric included in an SLA (e.g. availability, response time, elasticity, etc.). The formula has been applied to commercial SLAs (e.g. Amazon, Google, Microsoft) to demonstrate its applicability in terms of providers and different SLA metrics. Furthermore, the proposed formula / function and the corresponding complete SLA has been compiled into a machine-readable structural representation (i.e. JSON format) and a set of examples using this representation have also been developed.

The report is structured as follows: Section 2 provides the SLALOM proposed SLA specification, while Section 3 focuses on the SLA metric definition function and Section 4 presents concrete examples that showcase the applicability of the proposed function in different cases (both in terms of services and in terms of providers). Alignment with ISO and concrete examples through JSON representations are discussed in Section 5. Conclusions are drawn in section 6.

2 SLA Specification

This chapter proposes the SLALOM proposed specification / reference model building on top of the analysis performed with respect to standardization approaches and working groups outcomes, current SLAs offered by commercial cloud providers, expressed views by cloud providers and adopters, and research outcomes. This analysis was documented in the previous version of this report [1]. With respect to the ISO SLA working group outcomes, the SLALOM SLA specification in summary:

- Follows ISO 3534-2 [2] in terms of the expression of metrics in different scales (such as interval, ratio, nominal or ordinal).
- Follows ISO 19086-2 [3] in terms of the core blocks (i.e. metric definition, parameters definition, rule definition) and the corresponding elements (e.g. ID, name, unit, scale, etc.) of the SLA.
- Suggests changes to ISO 19086-2 for the naming of specific elements (e.g. referenceId is used both for metricId for parameterId and for ruleId, while SLALOM proposes the use of different identifiers).
- Suggests changes to ISO 19086-2 in order to include additional elements / components and blocks such as the dependency of a metric with other metrics (e.g. availability of storage service and dependency to latency or response time and dependency to bandwidth) and the importance of a metric comparing to other metrics included in an SLA.

2.1 Main SLA Building Blocks

Following the ISO 19086-2 SLA specification and the proposed SLA addition regarding the dependency block, the proposed building blocks of the SLALOM SLA specification / reference model are the following:

- *Metric:* The metric block corresponds to the service metric / objective (e.g. availability). Each metric is defined through standardized metric definitions, including the basic information that is necessary to understand the measurement of a property to be observed.
- *Parameter:* The parameter block links the metric with a set of parameters that need to be accompanied with the metrics (expressing in detail each metric). Parameters include how the metric has to be expressed (e.g. float, integer), what the customer should expect to observe from the specific metric of the SLA, and how different aspects quantify the corresponding metrics.
- *Rule*: The rule block refers to metric “constraints” (e.g. number of concurrent connections for a number of users metric), as elements that are used to further constrain some parts of each metric and indicate possible methods for measurement. Thus, for every metric there should be described its proposed generalized rules, including all the potential cases through, such as if/while statements, exponential increases in values, etc.
- *Dependency*: The dependency block is introduced by SLALOM and aims at capturing the dependencies between expressed metrics (e.g. response time and bandwidth).

2.2 SLA Components

For each one of the building blocks defined in the previous section, the corresponding elements / components are presented in this section.

2.2.1 Components used in all building blocks

There are specific components used for all building blocks, which refer to the following:

- *Identifier:* A unique identifier used for each metric, parameter, rule and dependency.
- *Name:* The name of the corresponding element in each block (e.g. name of a metric or name of a parameter, etc.).
- *Definition / Expression*: The definition (i.e. value) for each component / element. The definition can also be provided through an expression.
- *Unit*: The unit for the specific element (e.g. seconds, bytes, etc.).
- *Notes*: A component allowing the provision of notes for the specific element.

2.2.2 SLALOM proposed amendments

Taking into consideration the ISO 19086-2 SLA specification, SLALOM specification proposed the following changes:

- Addition of the *gradeOfImportance* component in the metric definition block, to define the metrics importance within an SLA if more than one metrics exist.
- Addition of the *consequenceOfViolation* component for the rule definition block, to define the potential consequence of violation on the service provisioning if the specific metric is violated.
- Use of *unique name for each identifier* to ensure their "uniqueness" instead of Id for all cases, thus SLALOM proposes to use metricId, parameterId, ruleId, and dependencyId.
- Removal of the *definition* component, as it is considered to be redundant taking into consideration the existence of the name and the note components of an SLA.

2.3 SLALOM SLA Specification / Reference Model

Based on the presented main building blocks (Section 2.1), the core components in each block (Section 2.2.1) and SLALOM proposed changes (Section 2.2.2) the proposed SLA Specification / Reference model is the following:

Table 1: SLALOM SLA Specification / Reference model

Metric	
metricId	A unique identifier for the metric
name	The name of the metric
unit	The unit that will be used for expressing the metric (e.g. seconds, bytes)
scale	Information on how the measurement value can be interpreted and what sort of operations can be performed on it (e.g. nominal, ordinal, interval, ratio)
gradeOfImportance	The grade of importance for the metric (values are integers in ascending

	order)
note	Formal and/or additional information related to the metric
Parameter	
parameterId	A unique identifier for the parameter
name	The name of the parameter
type	The type of the parameter definition, as for the way that it should be interpreted (e.g. integer, decimal, string, boolean, byte)
note	Formal and/or additional information related to the parameter
Rule	
ruleId	A unique identifier for the rule
name	The name of the rule
ruleExpression	The expression / function under which the specific metric of the SLA must obey
consequenceOfViolation	The consequence of violation of the rule / expression
note	Formal and/or additional information related to the rule
Dependency	
dependencyId	A unique identifier for the dependency
name	The name of the dependency
dependentMetricId	The unique identifier of the metric that the current metric has dependency with (in case of many dependencies, they should be split with a semicolon (;))
dependencyExpression	The expression under which the metric depends on another metric
note	Formal and/or additional information related to the dependency

3 SLA Metric Definition: Unified function

This chapter provides an abstract formal definition of a "metric" so as to provide the ground for a generic, yet uniform, definition of metrics. This function will be placed in the metric block of the SLA as described in Section 2.3. Based on this definition of a metric, the commercial SLAs of three different providers and three different types of services (IaaS, Storage and PaaS) are mapped to the SLALOM model proposal. Further representative examples for different metrics (i.e. availability, elasticity and response time) are provided as examples. The goal of such a unified function for a metric is to provide:

- Means to clarify any ambiguities that may exist in a definition of an SLA and prevent 3rd parties (e.g. SLA auditors) to accurately and non-repudiably monitor a given SLA
- A template from which providers may create instances of their respective SLAs
- Given that the template exists, automated or semi-automated tools may be created in order to create provider instances of SLAs
- Given that the metrics are expressed in a uniform manner in these instances, machine understandable representations may be acquired and processed automatically in order to export aspects such as provider rankings, comparisons etc.

3.1 Definition of abstract metric

The aim of the proposed definition is to enable any cloud provider to define a metric that will be included in the SLA. Given that in order to define and evaluate a metric, a set of samples against its validity are required, the proposed definition is directly linked to these samples. The inclusion of samples is strongly proposed in order to ensure that the metric is both clearly defined and can be evaluated with respect to its fulfilment. The latter has been inspired by the ambiguity that emerges from various existing SLAs such as Amazon EC2 [5] and Google Compute [7]. Based on the above, the abstract metric definition is achieved through three (3) individual levels / definitions: sample definition, boundary period and error definition, and abstract metric definition.

3.2 Sample definition

The aim of this level / definition is to enable the identification of the samples that satisfy a criterion related to their success. For example if availability is defined in a storage service not only in terms of the success of the operation but also with relation to performance aspects (e.g. GET operation of an object within "x" seconds), a success sample is the one for which the service responds within the "acceptable" time limit. Given that the samples are both of different nature and can be obtained through different mechanisms / means, the SLA specification (defined in the SLALOM SLA specification document in detail) should also include a "field / element" (in the "Rule Definitions" block) that concretizes the sampling process. This field, namely "*Type of operation*" will refer to the corresponding nature of the process.

The main argument for applying the Sample class, is the fact that it can lead directly to machine understandable descriptions, without the need for textual descriptions of rules. We demonstrate in the examples section sampling concretizations with the Sample class, without the need for text input. This

textual input would be even more difficult to be avoided in cases of more complex metrics, like in the cases of error ratios (GAE example that follows). Furthermore, with the inclusion of the way to acquire a sample, ambiguity in current SLAs with relation to the measurement process is avoided. The Sample class has the following attributes:

- Name: the label of the sample
- referenceId: the id of the sample definition
- timestamp: the time the sample was obtained, in whatever decided format e.g. ISO8601, ("2012-04-23T18:25:43.511Z"), identified as the JSON best practice
- scale: scale of the sample,
- value: the value of the sample. It must be stressed that the value may be either numeric or even textual, defined in the scale attribute (its usage as textual will be showed in the examples, especially GAE). Generic samples include only the obtained value, the limits for their incorporation are included in the next level expression in which the sample is used for the according metric or period calculation. Also the sample is a generic concept that may imply any way of obtaining information on a status or state of the service (e.g. email notification could also be considered as a way of being informed about the state).
- protocol: the protocol used to obtain the sample
- operation: name of the operation (e.g. type of method call, will be shown in the examples, especially Azure Storage SLA)

The following notation is used:

- *Sample Condition - sc*: the condition stating whether a sample has been successful.
    - operator: the operator can either be a boolean one (i.e. AND, OR, NOT) or a comparison operator (<, >, <=, >=, ==, !=).
    - value: the actual value of the condition that can be arithmetic, non-arithmetic (e.g. a string such as "exception") or an enumeration (e.g. HTTP response code == 200).
    - unit: the unit for the value of the condition.
- *Sample - s*: the sample used to evaluate a parameter against the condition sc.
- *Successful Sample - ss*: the sample satisfying the condition sc.
- *Unsuccessful Sample - us*: the sample not satisfying the condition sc.

Sample definition

For a given type of operation as specified in the corresponding field (described previously)

sc = operator + value + unit

ss = s if (sc is true)

us = s if (sc is false)

3.3 Boundary period and error definitions

The aim of this level / definition is to enable the definition of the boundary period and error for which the analysis of a parameter (through samples) is considered valid. The boundary period is the case for several providers today – for example Google sets a boundary condition to consider a downtime period

as actual downtime if it is larger than 5 consecutive minutes [6]. The same applies for error conditions. The overall goal of this level / definition is to identify the set of periods that are "valid" (as successful or unsuccessful) and should be included in the metric definition, based on the individual samples and the required error rate. The following notation is used:

- *Boundary Period - bp*: the period for which the analysis of a parameter (through samples) should be taken into account. Any sample that is not meeting this criterion (i.e. falls within the period) is excluded even though if it is successful (i.e. ss according to the sample definition).
    - operator: a comparison operator (<, >, <=, >=, ==, !=).
    - value: the actual arithmetic value of the condition.
    - unit: the unit in this case is always a time unit (e.g. seconds, minutes, etc).
- *Error Condition - ec*: the error condition ratio for which the analysis of a parameter (through samples) should be taken into account. The ratio is always expressed in a percentage (%) format.
    - operator: a comparison operator (<, >, <=, >=, ==, !=).
    - value: the actual arithmetic value of the condition.
- *Error Ratio - er*: the error ratio calculated based on the total set of samples and the successful samples.
- *Period - p*: the period in which samples (*sc* and *uc*) are examined according to the boundary period and the error condition.
- *Valid Period - vp*: the period for which the error ratio value meets the error condition ratio and the boundary period condition is also satisfied.
- *Non-valid Period - np*: the period for which the error ratio value does not meet the error condition ratio (the boundary period condition is satisfied).

Boundary period and error definitions

bp = operator + value + unit

ec = operator + value + %

er = Σus/Σs ∀us ∈ p

vp = p if ((er<=ec) && (p>=bp))

np = p if ((er>=ec) && (p>=bp))

3.4 Abstract metric definition

The aim of this level / definition is to provide an abstract format enabling cloud providers to define an SLA metric. While the definition is performed through a condition, a proposal is also provided linking the metric definition with each formal evaluation. The following notation is used:

- *Metric Condition - mc*: the condition regarding a specific metric. The condition is always expressed in a percentage (%) format to enable its evaluation as proposed through the metric evaluation.
    - operator: a comparison operator (<, >, <=, >=, ==, !=).
    - value: the actual arithmetic value of the condition.
- *Metric Evaluation - me*: the evaluation of the metric based on the valid and non-valid period samples. The evaluation should be smaller than the condition (i.e. me <mc).

Abstract metric definition
mc = operator + value + %
me = Σnp / (Σvp+Σnp)

The overall approach of the three layers appears in Figure 1.

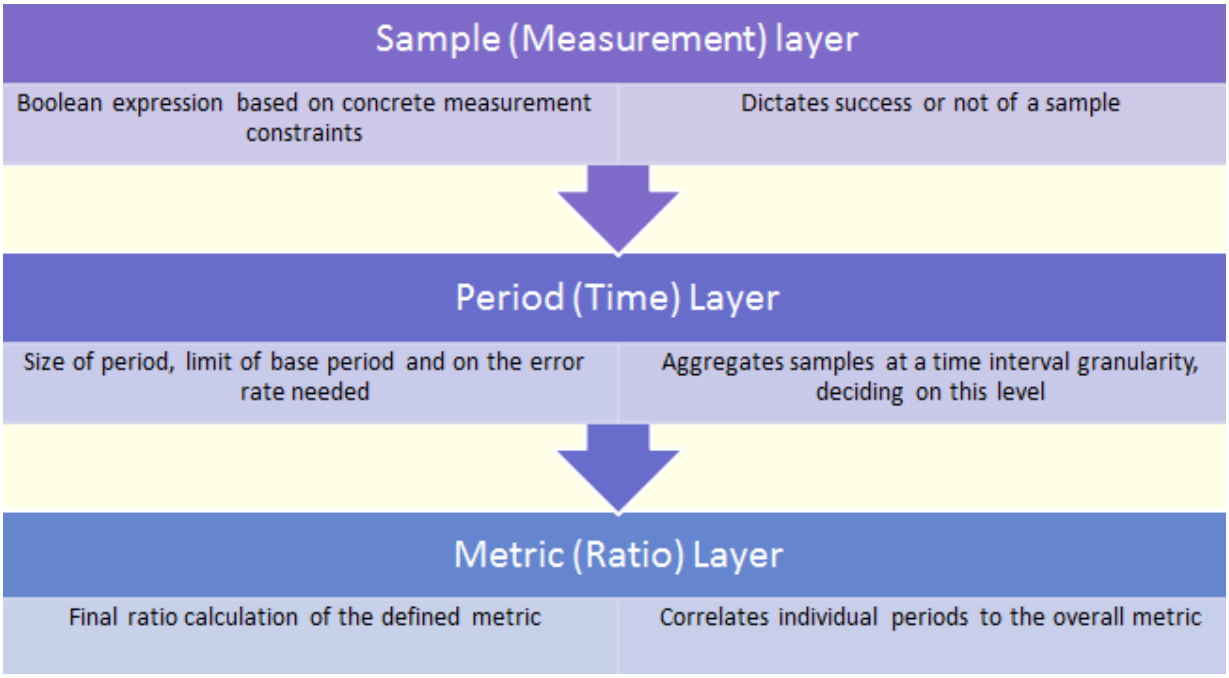


Figure 1: SLALOM proposed layer approach

4 Unified Function: SLA Examples

This section provides representative commercial examples of SLAs based on the aforementioned definitions in order to depict that the generic abstract definition can be adapted to different cases, providers and service types. There should be noted that for all examples the expressions that show validity (i.e. non violation) of the SLA are cited. The selection was performed based on different types of clauses that may exist, in order to demonstrate the applicability of the various layers.

4.1 Microsoft Azure Storage Services

The Azure Storage services SLA can be found in [4]. The most interesting aspect of the SLA in this case is the existence of a variety of conditions for identifying an error sample, including diversification in terms of the type of the operation performed. Another very interesting feature is the existence of concrete timing constraints for each such operation, a fact that merges performance aspects in the definition of availability.

Table 2: Microsoft Azure Storage Services Example

Microsoft Azure Storage		
Level / definition	Expression	Notes

Sample definition	sc = 2 sec	Several sampling conditions are defined per type of operation. For example it is specified (exact wording) "*Sixty (60) seconds*" for PutBlockList and GetBlockList.
	Type of operation: PutBlockList and GetBlockList	Several type of operations are defined. An example is provided here.
Boundary period and error definitions	bp > 3600 sec	The exact wording is "*given one-hour interval*".
	ec > 0%	Error condition reflecting that all periods should be taken into account for the availability metric evaluation (exact wording) "*is the sum of Error Rates for each hour*".
Abstract metric definition	availability < 99.9 %	Availability metric definition given the boundary period and error condition.

4.2 Amazon EC2 IaaS Services

This case represents probably the most typical SLA text that may be found in the domain [5]. It is mainly focused in the IaaS services domain and especially the VM level resources. The most interesting case here is the ambiguity that exists in how the availability of a sample is determined. While the text refers to "resources having external connectivity", it is not described how this is determined. In terms of networking, many different protocols may be used (some of them like ICMP by default disabled by Cloud providers due to potential security threats), while others may imply the existence of an application error residing in the VM that may be the reason of the failure (e.g. in the case of investigating whether a web server is up and running).

Table 3: Amazon EC2 Example

Amazon EC2		
Level / definition	Expression	Notes
Sample definition	sc: *UNDEFINED*	The sampling condition is not defined in the Amazon EC2 SLA. The concrete wording is "*when all of your running instances have no external connectivity*". Nonetheless, the way to specify / measure "external connectivity" is not defined. For example a customer could use a ping operation or a custom monitoring mechanism.
	Type of operation: *UNDEFINED*	Not defined how the condition of connectivity can be actually measured (e.g. the ping operation mentioned previously).
Boundary period and error definitions	bp > 60 sec	The exact wording is "the percentage of minutes", thus the period is 60 seconds.
	ec = 100%	Error condition reflecting that the error ratio is that for the entire bp the resource must be continuously "*unavailable*".

Abstract metric definition	availability < 99.95 %	Availability metric definition given the boundary period and error condition.

4.3 Google AppEngine Datastore Services

In this case the most interesting aspect is the coverage of the PaaS layer [6], as well as the existence of non-numerical expressions in order to identify the sample failure. Furthermore, one very interesting aspect is the existence of a minimum time interval in which the error must be continuously over a given threshold.

Table 4: Google AppEngine Datastore Example

Google AppEngine Datastore		
Level / definition	Expression	Notes
Sample definition	sc: INTERNAL_ERROR	Several sampling conditions are defined per type of operation. For example it is specified (exact wording) "*INTERNAL_ERROR, TIMEOUT, …*" for API calls.
	Type of operation: API calls	Several types of operations are defined. An example is provided here.
Boundary period and error definitions	bp > 300 sec	The exact wording is "*five consecutive minutes*".
	ec > 10%	Error condition reflecting that the error ratio is (exact wording) "*ten percent Error Rate*".
Abstract metric definition	availability < 99.95 %	Availability metric definition given the boundary period and error condition.

4.4 Additional examples

The purpose of this section is to depict the wide applicability of the proposed approach for the definition of SLA metrics both for different service classes (such as computational, storage and software services) and for different metrics (such as availability, elasticity and response time). After the examined commercial SLAs, we have also attempted to map the SLALOM layers to other types of metrics.

Table 5: Example for Availability for Storage Service

Availability for storage service		
Level / definition	Expression	Notes
Sample definition	sc <= 100 msec	Samples regarding availability obtained for example through ping operations to the corresponding hosts. Successful samples are the ones for which ping responds with less than 100 msec (above 100msec or "unreachable" are considered unsuccessful samples).
Boundary period and error definitions	bp > 300 sec	Boundary period of 300 secs reflecting that "sporadic" unavailability (based on the *sc*) will not be counted as actual unavailability periods.
	latency error ratio < 1%	Error condition ratio reflecting the number of cases

		for which latency (i.e. time for a single I/O operation) cannot exceed the specified value in the dependencyExpression SLA field (e.g. 50 msec).
Abstract metric definition	availability < 99.98 %	Metric definition with respect to availability given the boundary period and error condition (to be considered for the validation of the given availability constraint).

Table 6: Example for Elasticity for Computational Service

Elasticity for computational service		
Level / definition	Expression	Notes
Sample definition	sc <= 1 min	Samples regarding how fast the provider responds to requests for re-allocation of resources.
Boundary period and error definitions	bp > 10 min	Boundary period reflecting that non-allocation of some resources within 10 mins will not be counted as non-elasticity.
	ec < 5%	Error condition (precision) reflecting the number of resources deployed versus the actually needed ones.
Abstract metric definition	elasticity < 90 %	Metric definition with respect to elasticity given the boundary period and error condition.

Table 7: Example for Response Time for Software Service

Response time for software service		
Level / definition	Expression	Notes
Sample definition	sc <= 1 sec	Samples regarding response time obtained for through different requests (e.g. sequential, parallel, from different locations, etc). Either one or more than one sample conditions can be defined.
Boundary period and error definitions	bp < 30 sec	Boundary period of 30 secs reflecting for example the HTTP timeout period, within which requests not accommodated will not be counted as actual non-responsiveness.
	ec < 7%	Error condition (response) reflecting the number of cases for which the response time cannot exceed the specified value of the *sc*.
Abstract metric definition	response time < 97.77 %	Metric definition with respect to availability given the boundary period and error condition (to be considered for the validation of the given availability constraint).

5 Alignment with ISO baseline model

In this section we briefly explain the concept and elements of the ISO draft standard 19086-2 and then present in detail the SLALOM contribution to the ISO model. Finally, we highlight the SLALOM added value and differences to ISO approach by means of a comprehensive example.

5.1 ISO Baseline Work

5.1.1 Fields Definition

The ISO baseline model is portrayed in Figure 2. It foresees a number of fields to be populated, along with their names and conventions (more details to follow on the release of the draft standard 19086-2).

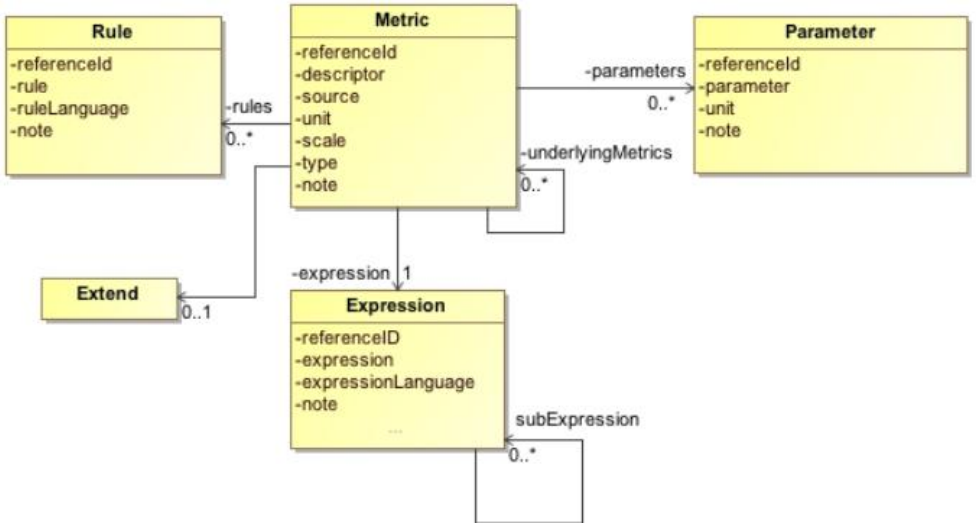


Figure 2: Baseline ISO model

The ISO model is an abstract model, meaning that it gives the choice to potential adopters to create descriptions in diverse ways, as long as they keep the naming conventions and fields usage, along with a set of minimal requirements about the content of specific fields.

5.1.2 Metrics and Underlying metrics concept

One of the most powerful features of the ISO initial draft of the model was the existence of the metrics field, which could have underlying metrics included in an iterative manner. This is especially helpful in the cases of more complex SLAs (such as the GAE and Microsoft azure) in which a top level metric is broken down to smaller intervals that have another metric in order to be decided if this baseline interval is to be included in the overall unavailable interval.

5.2 SLALOM Proposals and Extensions

5.2.1 Specific SLALOM approach on instantiation

The ISO model is an abstract model, meaning that it gives the choice to potential adopters to create descriptions in diverse ways, as long as they keep the naming conventions and fields usage. Thus no

specific requirement exists in how these fields will be expressed in order to map concepts that appear in real life SLAs.

As an example, an initial instantiation performed in ISO had the following form:

```
        unit="percentage">
        <Parameters>
            <Paramter name="billing cycle" referenceID="BP_001" unit="seconds"
note=""/>
        </Parameters>
        <Expression expression="CFA_002 = (BP_001 - UAP_001)/BC_001"
expressionLanguage="ISO8000"/>
        <UnderlyingMetrics>
            <MetricRef referenceID="UAP_001"/>
        </UnderlyingMetrics>
    </Metric>
    <Metric name="CloudServiceUnavailability" referenceID="UAP_001"
scale="RATIO" unit="seconds">
        <Expression expression="UAP_001=SUM(QDT_001)"
expressionLanguage="ISO8000"/>
        <UnderlyingMetrics>
            <MetricRef referenceID="QDT_001"/>
        </UnderlyingMetrics>
    </Metric>
    <Metric name="QualifiedDownTime" referenceID="QDT_001" scale="RATIO"
unit="seconds">
        <Rules>
            <Rule referenceID="QDT_R001"
                rule="The time duration starts when it is observed that the
cloud service is unavailable and ends when the cloud service is observed to be
not unavailable according to rule QDT_R002"/>
            <Rule referenceID="QDT_R002"
                rule=" a) The cloud service provides no response, or b)  the
cloud service returns a server error response to a valid user request during
two or more consecutive one minute intervals, or c) the cloud service fails to
deliver an average download time for a reference document of one second or
less. Unavailability due to scheduled maintenance is excluded from these
conditions and does not contribute towards unavailability calculations"/>
        </Rules>
        <Expression expression="delta(t)" expressionLanguage="ISO8000"/>
    </Metric>
</Main>
```

Figure 3: Initial generic instantiation example based on ISO model (Non SLALOM)

The circled part is the rules that govern some aspects of the measurement process. In this version of the instantiation, the normal text from the SLA is copied in the respective field, in order to fulfill the description needs. As can be easily understood, while this description follows the standard, it is not very useful in terms of machine readability or even comparability between SLAs. Text-based approaches imply the usage of Natural Language Processing techniques, which would great severe technical limitations. Furthermore, a number of critical aspects, such as the minimum failure interval that can be added in the general metric (defined in SLALOM as the boundary period), is not mapped to the definition, therefore it would not be taken under consideration by an adopter.

On the other hand, the SLALOM approach during instantiation included the following goals:

- Link to ***SLALOM layers as a mean to abstract and group similar concepts*** existing in various SLAs while using fields from the ISO baseline for standard compliance
- Minimize text thus enable maximum machine readability, ***through the translation of the SLA text to a set of parameters in the SLA, giving them numerical values or Boolean features***
- Usage and ***cross referencing of these parameters in the metrics sections*** in order to capture concepts such as minimum failure interval that should be used in the calculation
- ***Recursive usage of the (underlying) metric feature*** of the ISO baseline in order to indicate how it could be used in order to describe ***bottom layer features of the SLA and how these escalate towards the higher level metrics***
- Usage of the ***sample class in order to clear any practical ambiguities*** during the measurement process, as will be described in the following section

Concrete aspects of SLALOM instantiation appear in Section 5.3 while the step by step process is included in Section 7.1.

5.2.2 SLALOM Sample class extension

What was missing from the ISO baseline model was the ability to describe in a concrete, absolute and non-repudiable manner the way to perform measurements and obtain samples that would be used in order to calculate a respective (underlying or not) metric. SLALOM proposed the usage of an extension Class, i.e. the Sample class, that would capture all the needed information from a potential monitoring mechanism point of view, such as protocol used, limit of the operation etc. This extension enables full concretization of the SLA monitoring process and has been instantiated for three different cases of the SLA examples provided in Section 6. Furthermore, for each sample a cross-reference is made to which metric is used.

5.3 SLALOM compatible vs. ISO compatible SLAs

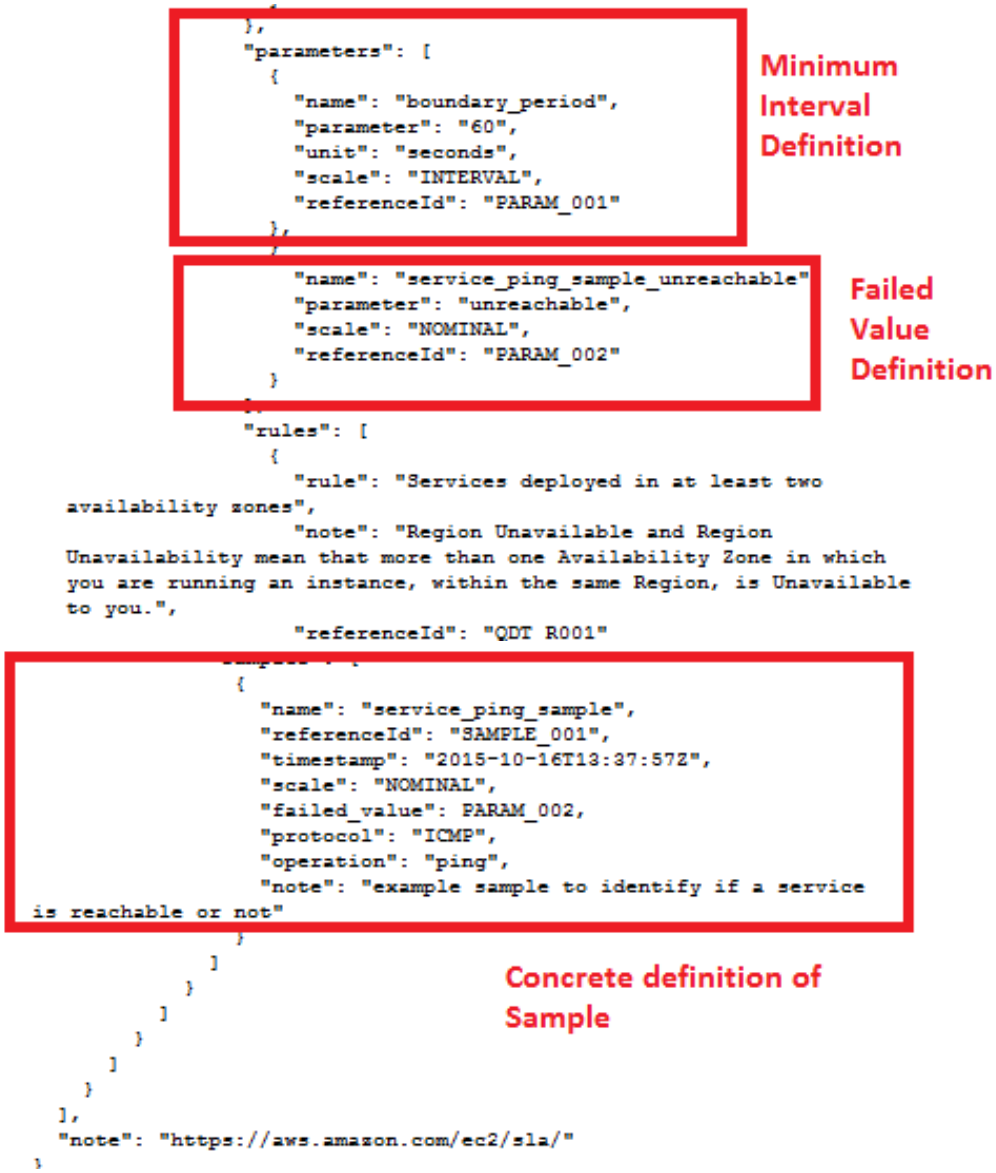
As a conclusion, by following the SLALOM instantiation process including the definition of the aforementioned parameters and their meaning, one may enable better machine readability and comparability of the respective fields and values, while maintaining alignment to the ISO standard. The SLALOM added value is evident if one compares the outcome of our modeling of AWS EC2 to that of Figure 3, which shows an ISO-compatible non-SLALOM approach.

The following example presents the SLALOM modeling and highlights the differences with the aforementioned non-SLALOM version.


```
{
  "name": "AWS_SLA_violation",
  "referenceId": "ASV_001",
  "scale": "NOMINAL",
  "expression": {
    "expression": "CFA_002<PARAM_002",
    "expressionLanguage": "ISO80000"
  },
  "parameters": [
    {
      "name": "availability_limit",
      "referenceId": "PARAM_002",
      "unit": "%",
      "parameter": "99.95"
    }
  ],
  "underlyingMetrics": [
    {
      "name": "CloudServiceAvailability",
      "referenceId": "CFA_002",
      "unit": "%",
      "scale": "RATIO",
      "expression": {
        "expression": "CFA_002 = ((BP_001 - UAP_001) / BC_001)>
PARAM_002",
        "expressionLanguage": "ISO80000"
      },
      "parameters": [
        {
          "name": "billing cycle",
          "referenceId": "BP_001",
          "unit": "month",
          "parameter": "1"
        }
      ],
      "underlyingMetrics": [
        {
          "name": "CloudServiceUnavailability",
          "referenceId": "UAP_001",
          "unit": "second",
          "scale": "RATIO",
          "expression": {
            "expression": "UAP_001 = SUM(QDT_001)",
            "expressionLanguage": "ISO80000"
          },
          "underlyingMetrics": [
            {
              "name": "CloudServiceUnavailability_INTERVAL",
              "referenceId": "QDT_001",
              "unit": "second",
              "scale": "INTERVAL",
              "expression": "IF (QDT_001_TEMP > PARAM_001) THEN
QDT_001 = QDT_001_TEMP",
              "expressionLanguage": "ISO80000",
              "subExpressions": [
                {
                  "expression": "IF (SAMPLE_001 = PARAM_002) THEN
QDT_001_TEMP = delta(SAMPLE_001.timestamp)",
                  "expressionLanguage": "ISO80000"
                }
```


Parameterized Value of Overall Metric

BaseInterval Calculation



Some insight into how this process was performed or how it can be exploited is portrayed in section 7.

Benefits achieved from the specific process include:

- Achieve SLA **non-repudiation**, so that the measurement cannot be contested
- Establish **trust** and **transparency** for service execution compliant to the terms and proper **violation management**
- Enable **automation** of contract and performance management and monitoring
- Aid the involvement of actors like **trusted third parties** offering relevant services

The following table summarizes the main differences between ISO compliant and SLALOM compliant SLAs.

Table 8: Main differences between ISO-compliant and SLALOM-compliant SLAs

ISO compliant SLA	SLALOM compliant SLA
• Usage of the ISO fields (classes, parameters) • SLA not necessarily fully defined	• ISO compliant • Clear and Well-defined • Non-repudiable • Machine understandable • Facilitates comparability

6 ISO-Aligned SLALOM Model: Machine understandable examples

Following the cooperation with the ISO baseline model, the aforementioned (see Section 4) commercial examples have been reformulated to include the joint approach, including the SLALOM based extensions that are necessary in order to unambiguously declare the SLA parameters in a machine understandable case. The results are included in the following JSON descriptions for the identified original examples. With relation to the Rules field, we propose the strict definition of the Rules class to be concerning the necessary preconditions to apply for a given deployment to be eligible for an SLA. Example rules of this case may include, based on a given SLA:

- Deployment in different Availability Zones
- Enablement of specific features like replication options
- Throttling of requests in case of unavailability
- Scheduled Maintenance Downtime
- etc.

Given that all concepts are depicted without the need of text, we may use the Note field as an informative placeholder of the relevant SLA text that dictated the specific section creation.

6.1 JSON Implementation Instance – Extended Simplified Model for AWS EC2 (IaaS)


```
{
  "name": "AWS_SLA_violation",
  "referenceId": "ASV_001",
  "scale": "NOMINAL",
  "expression": {
    "expression": "CFA_002<PARAM_002",
    "expressionLanguage": "ISO80000"
  },
  "parameters": [
    {
      "name": "availability_limit",
      "referenceId": "PARAM_002",
      "unit": "%",
      "parameter": "99.95"
    }
  ],
  "underlyingMetrics": [
    {
      "name": "CloudServiceAvailability",
      "referenceId": "CFA_002",
      "unit": "%",
      "scale": "RATIO",
      "expression": {
        "expression": "CFA_002 = ((BP_001 - UAP_001) / BC_001)>PARAM_002",
        "expressionLanguage": "ISO80000"
      },
      "parameters": [
        {
          "name": "billing cycle",
          "referenceId": "BP_001",
          "unit": "month",
```



```
          "parameter": "1"
        }
      ],
      "underlyingMetrics": [
        {
          "name": "CloudServiceUnavailability",
          "referenceId": "UAP_001",
          "unit": "second",
          "scale": "RATIO",
          "expression": {
            "expression": "UAP_001 = SUM(QDT_001)",
            "expressionLanguage": "ISO80000"
          },
          "underlyingMetrics": [
            {
              "name": "CloudServiceUnavailability_INTERVAL",
              "referenceId": "QDT_001",
              "unit": "second",
              "scale": "INTERVAL",
              "expression": {
                "expression": "IF (QDT_001_TEMP > PARAM_001) THEN QDT_001 =
QDT_001_TEMP",
                "expressionLanguage": "ISO80000",
                "subExpressions": [
                  {
                    "expression": "IF (SAMPLE_001 = PARAM_002) THEN QDT_001_TEMP =
delta(SAMPLE_001.timestamp)",
                    "expressionLanguage": "ISO80000"
                  }
                ]
              },
              "parameters": [
                {
                  "name": "boundary_period",
                  "parameter": "60",
                  "unit": "seconds",
                  "scale": "INTERVAL",
                  "referenceId": "PARAM_001"
                },
                {
                  "name": "service_ping_sample_unreachable",
                  "parameter": "unreachable",
                  "scale": "NOMINAL",
                  "referenceId": "PARAM_002"
                }
              ],
              "rules": [
                {
                  "rule": "Services deployed in at least two availability zones",
                  "note": "Region Unavailable and Region Unavailability mean that more
than one Availability Zone in which you are running an instance, within the same
Region, is Unavailable to you.",
                  "referenceId": "QDT_R001"
                }
              ],
              "samples": [
                {
                  "name": "service_ping_sample",
                  "referenceId": "SAMPLE_001",
                  "timestamp": "2015-10-16T13:37:57Z",
                  "scale": "NOMINAL",
                  "value": "unreachable",
                  "protocol": "ICMP",
```



```
                    "operation": "ping",
                    "note": "example sample to identify if a service is reachable or
not"
                  }
                ]
              }
            ]
          }
        ]
      }
  ],
  "note": "https://aws.amazon.com/ec2/sla/"
}
```

6.2 JSON Implementation Instance – Extended Simplified Model for Microsoft Azure SLA (Storage services)

```
{
  "name": "Microsoft Azure Storage SLAViolation",
  "referenceId": "MAS_001",
  "scale": "NOMINAL",
  "expression": {
    "expression": "CFA_002 < PARAM_002",
    "expressionLanguage": "ISO80000"
  },
  "parameters": [
    {
      "name": "availability_limit",
      "referenceId": "PARAM_002",
      "unit": "%",
      "parameter": "99.9"
    }
  ],
  "underlyingMetrics": [
    {
      "name": "Monthly Uptime Percentage",
      "referenceId": "CFA_002",
      "unit": "%",
      "scale": "RATIO",
      "expression": {
        "expression": "CFA_002 = 100 - AER_001",
        "expressionLanguage": "ISO80000"
      },
      "underlyingMetrics": [
        {
          "name": "Average Error Rate",
          "referenceId": "AER_001",
          "unit": "%",
          "scale": "RATIO",
          "expression": {
            "expression": "AER_001 = SUM(HER_001) AND HER_001 belonging to BP_001",
            "expressionLanguage": "ISO80000"
          },
          "parameters": [
            {
              "name": "billing cycle",
              "referenceId": "BP_001",
              "unit": "month",
              "parameter": "1"
            }
          ],
          "underlyingMetrics": [
```

```
            {
              "name": "Hourly Error Rate",
              "referenceId": "HER_001",
              "unit": "%",
              "scale": "RATIO",
              "expression": {
                "expression": "HER_001=HER_003/HER_002",
                "expressionLanguage": "ISO80000",
                "subExpressions": [
                  {
                    "expression": "HER_002=SUM(SAMPLE_001 belonging to PARAM_001)",
                    "expressionLanguage": "ISO80000",
                    "note": "Number of samples within the boundary period"
                  },
                  {
                    "expression": "HER_003=SUM(SAMPLE_001.value > PARAM_003 belonging
to PARAM_001)",
                    "expressionLanguage": "ISO80000",
                    "note": "Number of error samples within the boundary period"
                  }
                ]
              },
              "parameters": [
                {
                  "name": "boundary_period",
                  "parameter": "3600",
                  "unit": "seconds",
                  "referenceId": "PARAM_001"
                },
                {
                  "name": "GET BLOCK LIST LIMIT",
                  "value": "60",
                  "unit": "seconds",
                  "referenceId": "PARAM_003"
                },
                {
                  "name": "billing cycle",
                  "referenceId": "BP_001",
                  "unit": "month",
                  "parameter": "1"
                }
              ],
              "samples": [
                {
                  "name": "STORAGE GET BLOCK LIST API CALL response time",
                  "referenceId": "SAMPLE_001",
                  "timestamp": "2015-10-16T13:37:57Z",
                  "scale": "interval",
                  "value": "49",
                  "unit": "seconds",
                  "protocol": "REST",
                  "operation": "GetBlockList",
                  "note": "example sample to measure the response time of the service"
                }
              ]
            }
          ]
        }
      ]
    }
  ],
  "note": "https://azure.microsoft.com/en-us/support/legal/sla/storage/v1_0/"
}
```

6.3 JSON Implementation Instance – Extended Simplified Model for GAE Datastore (PaaS)


```
{
  "name": "GAE_SLA_violation",
  "referenceId": "ASV_001",
  "unit": "",
  "scale": "NOMINAL",
  "expression": {
    "expression": "CFA_002<PARAM_002",
    "expressionLanguage": "ISO80000"
  },
  "parameters": [
    {
      "name": "availability_limit",
      "referenceId": "PARAM_002",
      "unit": "%",
      "scale": "RATIO",
      "parameter": "99.95"
    }
  ],
  "underlyingMetrics": [
    {
      "name": "CloudServiceAvailability",
      "referenceId": "CFA_002",
      "unit": "%",
      "scale": "RATIO",
      "expression": {
        "expression": "CFA_002 = ((BP_001 - UAP_001) / BP_001)",
        "expressionLanguage": "ISO80000"
      },
      "parameters": [
        {
          "name": "billing cycle",
          "referenceId": "BP_001",
          "unit": "month",
          "scale": "INTERVAL",
          "parameter": "1"
        }
      ],
      "underlyingMetrics": [
        {
          "name": "CloudServiceUnavailability",
          "referenceId": "UAP_001",
          "unit": "second",
          "scale": "RATIO",
          "expression": {
            "expression": "UAP_001 = SUM(QDT_001)",
            "expressionLanguage": "ISO80000"
          },
          "underlyingMetrics": [
            {
              "name": "CloudServiceUnavailability_INTERVAL",
              "referenceId": "QDT_001",
              "unit": "second",
              "scale": "INTERVAL",
              "expression": {
                "expression": "QDT_001 = IF (DUR_001 > PARAM_001 AND ER_001 >
PARAM_002) THEN QDT_001 = DUR_001",
                "expressionLanguage": "ISO80000",
                "subExpressions": [
                  {
                    "expression": "DUR_001 = delta(SAMPLE_001.timestamp)",
                    "expressionLanguage": "ISO80000"
```

```
                    },
                    {
                      "expression": "ER_001=SUM(SAMPLE_001.value belonging to
PARAM_003)/SUM(SAMPLE_001)",
                      "expressionLanguage": "ISO80000"
                    }
                  ]
                },
                "parameters": [
                  {
                    "name": "boundary_period",
                    "parameter": "300",
                    "unit": "seconds",
                    "scale" : "INTERVAL",
                    "referenceId": "PARAM_001"
                  },
                  {
                    "name": "error_rate",
                    "parameter": "10",
                    "unit": "%",
                    "scale" : "RATIO",
                    "referenceId": "PARAM_002"
                  },
                  {
                    "name": "SLA VIOLATION API RESPONSES",
                    "parameter": [
                      "INTERNAL_ERROR",
                      "TIMEOUT",
                      "BIGTABLE_ERROR",
                      "COMMITTED_BUT_STILL_APPLYING",
                      "TRY_ALTERNATE_BACKEND"
                    ],
                    "scale": "NOMINAL",
                    "referenceId": "PARAM_003"
                  }
                ],
                "samples": [
                  {
                    "name": "datastore_API_CALL",
                    "referenceId": "SAMPLE_001",
                    "timestamp": "2015-10-16T13:37:57Z",
                    "scale": "NOMINAL",
                    "value": "a response value string",
                    "protocol": "REST",
                    "operation": "API CALL",
                    "note": "example sample to identify the service response status"
                  }
                ]
              }
            ]
          }
        ]
      }
    ],
    "note": "https://cloud.google.com/appengine/sla?hl=en"
}
```

7 Guidelines to Adopters

With relation to potential adopters, we foresee the following roles involved in the process of SLA definition or usage:

- Cloud providers, that may need to map their SLA to the standardized description, or hired third parties, e.g. consultants, to perform this process on their behalf;
- Cloud users, that may need to understand and compare different providers, select one and then monitor their running services or hired third parties, e.g. consultants, to perform this process on their behalf.

These are depicted in the following UML use case diagram (Figure 4).

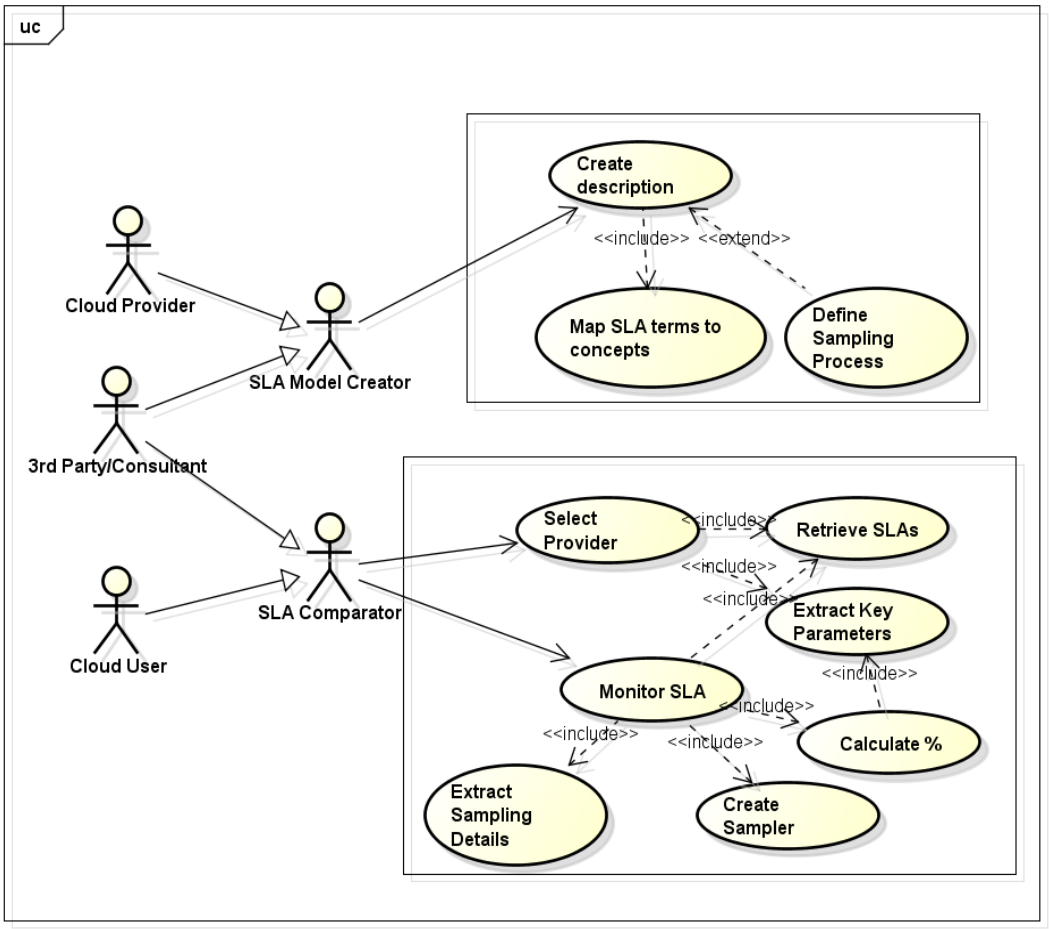


Figure 4: Potential adopters use cases

7.1 Model Creators/Instantiators as Adopters

As mentioned in Section 5.2.1, SLALOM offers a unique approach on the instantiation process, with the aforementioned goals and benefits. In this section we aim to provide a practical guide as to how this can

be implemented in order to cover the goals of machine readability and comparability, mapping the existing SLA text to parameters.

As an example, we follow a step by step process in order to produce the Google App Engine Data Store SLA, a PaaS layer SLA that portrays a number of difficulties and specific clauses. For each step we also include the respective SLA text that instructs the concretization of a specific parameter.

7.1.1 Parameter and Metric Hierarchy extraction

Initially the Model Creator needs to go through the overall SLA text and highlight the aspects that should be described. Where applicable, they should have under consideration the respective defined concepts of SLALOM, such as the bounding period and the error rate.

This process for the GAE Datastore SLA [6] appears in Figure 5.

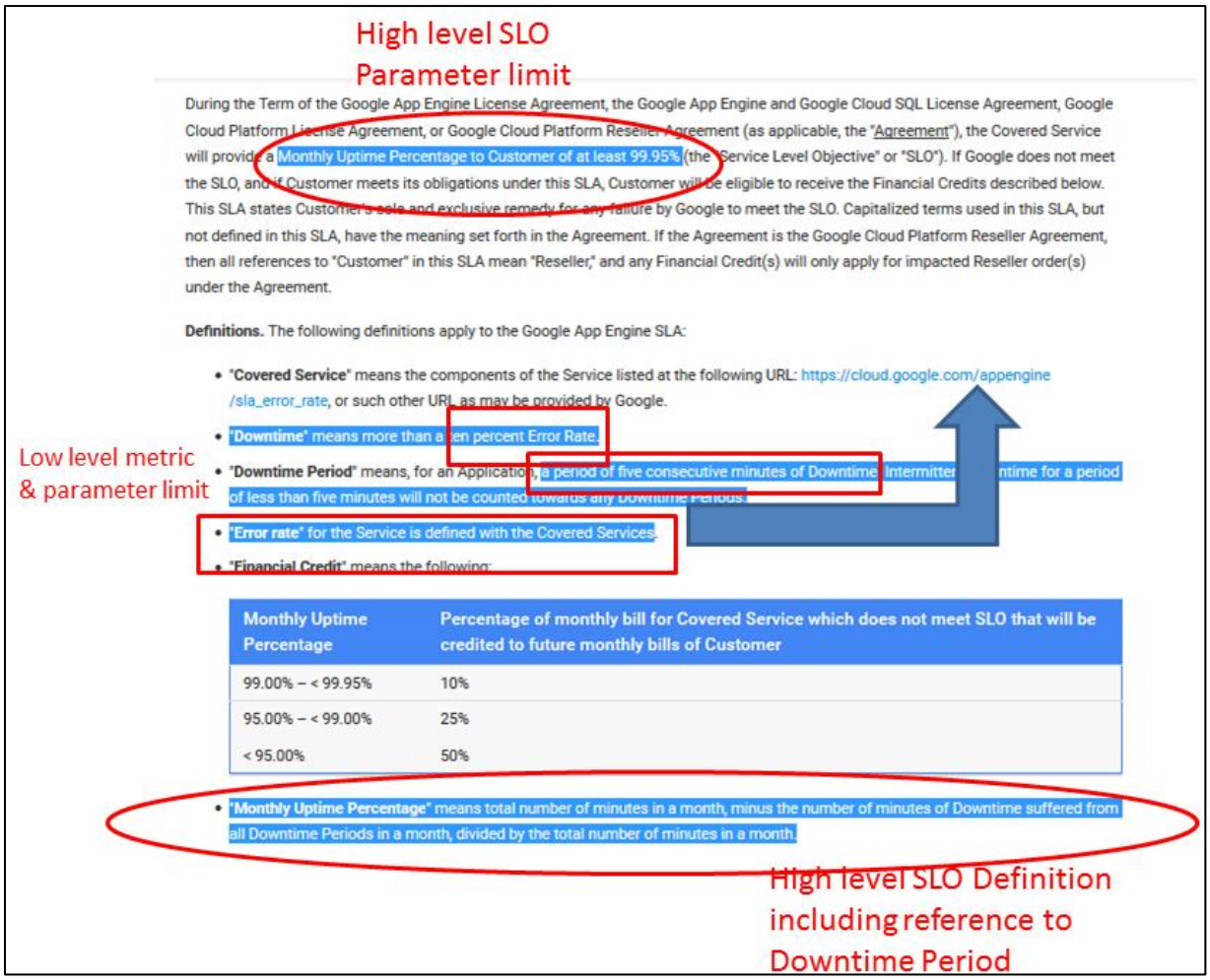


Figure 5: GAE Datastore SLA text

The highlighted points are the ones that portray interest. In a top-down approach, the SLA starts by describing the high level SLO, denoted as Monthly Uptime Percentage, giving directly its limit. This indicates that a parameter with a value of 99.95 should be defined at the same description level (top

level) as the MUP. Concrete wording is “The Covered Service will provide a Monthly Uptime Percentage to Customer of at least 99.95% (the "Service Level Objective" or "SLO").”

The extracted first part appears in the following figure, including the definition of GAE SLA violation to be based on MUP and to be less than 99.95% in order to have a violation.


```
{
  "name": "GAE_SLA_violation",
  "referenceId": "ASV_001",
  "unit": "",
  "scale": "NOMINAL",
  "expression": {
    "expression": "MUP<PARAM_002",
    "expressionLanguage": "ISO80000"
  },
  "parameters": [
    {
      "name": "availability_limit",
      "referenceId": "PARAM_002",
      "unit": "%",
      "scale": "RATIO",
      "parameter": "99.95"
    }
  ],
```


Now we need to actually define the MUP metric. The actual definition of the MUP appears at the bottom of the page (second red circle of Figure 5), including the details about its calculation that refer to the underlying metric of Downtime, which in turn is defined through Downtime periods (another underlying metric). The concrete wording is *“’Monthly Uptime Percentage’ means total number of minutes in a month, minus the number of minutes of Downtime suffered from all Downtime Periods in a month, divided by the total number of minutes in a month.”*.

Thus from this information we can extract the high level metric, the calculation formula for MUP and the needed hierarchy as follows:

- Violation level (top metric)
    - MUP (underlying metric of violation)
        - DownTime (underlying metric of MUP)
            - DownTime Period (underlying metric of Downtime, calculated based on samples and constraints of this level)
                - Sample (bottom class)

For the MUP level, and based on its definition, the description includes only a link to the Downtime UAP_001, mapping the wording to the expression field. Another parameter is the billing cycle of 1 month, which is extracted from the text wording “*divided by the total number of minutes in a month*”. The Downtime UAP_001 is then defined as the sum of the downtime periods, in the underlying metric definition of UAP.


```
"name": "CloudServiceAvailability",
"referenceId": "MUP",
"unit": "%",
"scale": "RATIO",
"expression": {
  "expression": "MUP = ((BP_001 - UAP_001) / BP_001)",
  "expressionLanguage": "ISO80000"
},
"parameters": [
  {
    "name": "billing cycle",
    "referenceId": "BP_001",
    "unit": "month",
    "scale": "INTERVAL",
    "parameter": "1"
  }
],
"underlyingMetrics": [
  {
    "name": "CloudServiceUnavailability",
    "referenceId": "UAP_001",
    "unit": "second",
    "scale": "RATIO",
    "expression": {
      "expression": "UAP_001 = SUM(QDT_001)",
      "expressionLanguage": "ISO80000"
```

Wording: total number of minutes in a month, minus the number of minutes of Downtime suffered from all Downtime Periods in a month, divided by the total number of minutes in a month

So now we need to continue the definition about the Downtime period (denoted as Low level metric and parameter limit in Figure 5). From the SLA wording the respective text includes:

- *"Downtime" means more than a ten percent Error Rate.*
- *"Downtime Period" means, for an Application, a period of five consecutive minutes of Downtime. Intermittent Downtime for a period of less than five minutes will not be counted towards any Downtime Periods.*
- *"Error rate" for the Service is defined with the Covered Services.*

From this we can conclude that for a baseline qualified downtime interval to be calculated we need to have both requirements addressed (ER>10%-PARAMETER, and interval>5 minutes->300 seconds-PARAMETER). Thus two new parameters need to be defined:

```
"parameters": [
  {
    "name": "boundary_period",
    "parameter": "300",
    "unit": "seconds",
    "scale" : "INTERVAL",
    "referenceId": "PARAM_001"
  },
  {
    "name": "error_rate",
    "parameter": "10",
    "unit": "%",
    "scale" : "RATIO",
    "referenceId": "PARAM_002"
  },
```


One elementary calculated QDT interval is defined as follows, mapping to the two requirements expression:

```
"underlyingMetrics": [
  {
    "name": "CloudServiceUnavailability_INTERVAL",
    "referenceId": "QDT_001",
    "unit": "second",
    "scale": "INTERVAL",
    "expression": {
      "expression": "QDT_001 = IF (DUR_001 > PARAM_001
AND ER_001 > PARAM_002) THEN QDT_001 = DUR_001",
      "expressionLanguage": "ISO80000",
      "subExpressions": [
        {
          "expression": "DUR_001 = delta(SAMPLE_
001.timestamp)",
          "expressionLanguage": "ISO80000"
        },
        {
          "expression": "ER_001=SUM(SAMPLE_001.value
belonging to PARAM_003)/SUM(SAMPLE_001)",
          "expressionLanguage": "ISO80000"
        }
      ]
```

One aspect that we have to describe is the two sub expressions in the previous figure. The duration is based on a timestamp value of a Sample class. The sample class is necessary in order to describe individual samples and is defined as follows:

```
"samples": [
  {
    "name": "datastore_API_CALL",
    "referenceId": "SAMPLE_001",
    "timestamp": "2015-10-16T13:37:57Z",
    "scale": "NOMINAL",
    "value": "a response value string",
    "protocol": "REST",
    "operation": "API CALL",
    "note": "example sample to identify the service
response status"
  }
```

This indicates that as sample we denote any REST-based API call made towards the GAE Datastore engine. The timestamp of this call may be used in order to calculate the duration of a QDT interval, if the samples follow also the necessary Error rate condition. The Error Rate is defined as the ratio of these API calls, for which a specific error response is given. This hidden definition is included in the link made available to the Covered service field of Figure 5 and includes the specific responses that count as errors. These need also to be defined in the SLA description, in order to have the full details, as follows:


```
{
  "name": "SLA VIOLATION API RESPONSES",
  "parameter": [
    "INTERNAL_ERROR",
    "TIMEOUT",
    "BIGTABLE_ERROR",
    "COMMITTED_BUT_STILL_APPLYING",
    "TRY_ALTERNATE_BACKEND"
  ],
  "scale": "NOMINAL",
  "referenceId": "PARAM_003"
}
```


Therefore, if we make a number of API calls to the service and the ratio of replies has the aforementioned responses in a parameter > 10% for more than 5 minutes, this interval may be calculated (along with all other similar intervals of the month) towards the overall MUP calculation.

Compensation is not covered within this case since it is not part of the concrete metric definition.

7.2 Model comparators as Adopters

Once the SLA descriptions are in place, a Model comparator may retrieve these and engage in a selection and/or monitoring process of the eventually selected provider.

For the selection and monitoring process, we anticipate that comparisons will be made against SLAs of similar layers (e.g. IaaS vs. IaaS, PaaS vs. PaaS etc.). In such an attempt, having a few pointers to perform the process is key for understanding how such a comparison may be made:

- A parameter mapping to numerical values similar to the one presented in [8]. However one needs to be careful in this process since a number of parameters may or may not apply in some cases.
- Parameters such as the bounding period are critical for understanding the differences between provider definitions. Especially if one is interested in a continuous service with high degrees of interactivity, having small intervals of unavailability might be a deal breaker. However typically these small intervals do not count towards the overall SLA violation percentage due to the boundary period used by the providers.
- Understand the meaning and values of the parameters from the previous section, how to obtain them programmatically (which fields to filter) or directly compare them visually.
- How to map them to your interests via tailor made comparisons: if someone is interested in running full-time services they do not care about the difference in the monthly cycle vs actual usage time of the service feature.
- Do not consider the actual values for availability only, since the general case is that these are not compatible between providers (thus while the value might be the same, e.g. 99.95%, it is extracted with different formulas).
- Consider creating a software component that directly retrieves the available descriptions and ranks them. Through the SLALOM instantiation way, with the extended usage of numerical

values, this automated approach is a real enabler for doing generic comparisons without being caught in the frenzy of marketing numbers.

- Once the selection is finished, monitoring of an SLA is the next step. Through the usage of the Sample class definition by the providers, a potential SLA auditor will be in an ideal state to create software agents that may complete the task. Thus selecting providers that have used such Sample classes is key also for this stage, in order to avoid unnecessary confrontation. Furthermore, any auditing attempt should be completely adapted to the formulas and parameters defined by the provider.

8 Conclusions

Following the analysis documented in the first document of this series of deliverables, this (second edition) report provides the proposed SLALOM SLA specification / reference model, following and extending the under-development ISO specification. Furthermore, the report provides a function enabling the definition of any metric from providers in a unified way. Examples showcasing its applicability, as well as JSON representative examples of the complete specification are also provided. Furthermore, concrete guidelines are provided for prospective model adopters. It should be noted that the SLALOM proposals have also been submitted to the ISO WG in order to be included (if accepted) in the ISO SLA specification.

Future work includes finalization of the proposed SLA specification and inclusion of legal and privacy aspects both in the SLA specification and in the SLA lifecycle process through the corresponding mechanisms that will be proposed.

9 References

[1] SLALOM SLA Specification and Reference Model – a – (Public Deliverable), available at: http://slalom-project.eu/content/d32-%E2%80%93-sla-specification-and-reference-model

[2] ISO 3534-2, Statistics - Vocabulary and symbols - Part 2: Applied statistics

[3] ISO/IEC 19086-2, Information Technology - Cloud Computing - Service Level Agreement (SLA) Framework and Terminology - Part 2: Metrics

[4] Microsoft Azure Storage SLA text, available at: https://azure.microsoft.com/en-us/support/legal/sla/storage/v1_0/

[5] Amazon EC2 Service Level Agreement, available at: https://aws.amazon.com/ec2/sla/

[6] Google App Engine Service Level Agreement, available at: https://cloud.google.com/appengine/sla

[7] Google Compute Engine Service Level Agreement, available at: https://cloud.google.com/compute/sla

[8] Nikolas Herbst, Rouven Krebs, Giorgos Oikonomou, George Kousiouris, Athanasia Evangelinou, Alexandru Iosup, Samuel Kounev: Ready for Rain? A View from SPEC Research on the Future of Cloud Metrics, available at: https://research.spec.org/fileadmin/user_upload/documents/rg_cloud/endorsed_publications/SPEC-RG-2016-01_CloudMetrics.pdf

10 Glossary of Acronyms

Acronym	*Definition*
Amazon EC2	Amazon Elastic Compute Cloud
Amazon EBS	Amazon Elastic Block Size
Amazon S3	Amazon Simple Storage Service
AWS	Amazon Web Service
C-SIG	Cloud Select Industry Group
CSP	Cloud Service Provider
EEA	European Economic Area
EFTA	European Free Trade Association
EU	European Union
IaaS	Infrastructure as a Service
MSA	Master Service Agreement
PaaS	Platform as a Service
PWS	Pivotal Web Service
SaaS	Software as a Service
SLA	Service level Agreement
SLO	Service level Objective
SQO	Service Qualitative Objective

