tmforum
inform innovate accelerate optimize

Frameworx Specification

SLA Management API REST Specification

Digital Ecosystem Suite
TMF623
October 2013

Latest Update: Frameworx Release 13.5	**Member Evaluation**
Version 0.7.1	**IPR Mode: RAND**

NOTICE

Direct inquiries to the TM Forum office:

240 Headquarters Plaza,
East Tower – 10th Floor,
Morristown, NJ 07960 USA
Tel No. +1 973 944 5100
Fax No. +1 973 944 5110
TM Forum Web Page: www.tmforum.org

TABLE OF CONTENTS

LIST OF TABLES

No table of figures entries found.

INTRODUCTION

The following document is the specification of the REST API for the SLA and SLA Violation resources. It includes the model definition as well as all available operations. Possible actions are creating and retrieving a SLA or SLA Violation, updating the whole SLA or only do a patch update. Furthermore the HTTP GET allows filtering.

RESOURCE MODEL

SLA

Summary

In this specification we are addressing only two SLA Resource Model “SLA” and “SLA Violation”, in one hand, and “SLAViolationEvent” as a notification, in the other hand.

From operation perspective, we identified which operations (GET, PUT, PATCH, and POST) are relevant, required, needed for Hackathon or to be considered in a future release in order to be applied to these resources alongside the notification.

a) SLA resource model Description

This SLA Resource Model aims at illustrating the way to translating “SLA Resource Model” in JSON representation, reflecting the resource aspect described in the “SLA API Profile” (Management Requirements) document V0.2. Interaction between the resources as described in SID SLA modeling is a main focus in this JSON representation work. Besides, regarding “SLA States” modeling, we used “WS-Agreement” diagram as recommended in the “SLA Profile” document (section 4) when applicable.

For instance, we don’t represent the “SLA Agreement” as a standalone object; rather we rely on existing SID “Agreement”, “Agreement items” and its interactions, meaning SLA “Approval” and “Terms or Conditions” and “Consequence” processes. All these processes that are linked to the “Negotiation Phase” of the SLA Management lifecycle are assumed to be completed. That means, we assume only one state “Observed” which supposes the SLA is approved and accepted by the involved parties. The latter are represented in terms of roles, each with its rights & duties covering Multi-partner model. Here we are illustrating B2B model (SLA Provider, SLA Consumer, SLA Auditor, EndUser roles). In our model, “SLA Provider” is referring to a CSP while “SLA Consumer” is referring to a DSP. The “EndUser” is referring to the customer of the DSP, in some cases he could be a customer of both (CSP and DSP). “SLA Auditor” role is to monitor SLA as described in TR178V2. It could be played either by the CSP himself or delegated to a 3rd party. The “Agreed” or “Approved SLA” is described in terms of “SLS” which contains the KPIs / KQIs, their related values or range, thresholds, valid period or date, consequences in case of violation of any clause of the SLA. We considered a “Single” SLS model as well as “Composite” SLS one.

We also assume all KPIs (Metrics) are existing ones, stored in the Service Provider “Metrics Library” with their attached references. Besides, each metric is attached to a given Product in the Catalogue we can refer to via an ”URL”. We also characterized the SLA resource model by its “validly period”. This use case covers the situation where the validity period is predetermined (planned) which excludes the case of Time-variant SLA that could be attached to a “SLA on-Demand” use case. The latter could be considered in another release for specific use cases (Cloud, virtualization) for instance.

In order to optimize this SLA resource model, we tried to define a common pattern or Template for the SLS we named “rule”. Indeed, this “rule” intends to reflect prescribed items a SLS may contain. So, this pattern is structured as following: the Id of the metric, its name, the measurement unit attached to the considered metric, its reference value, the tolerance value when the threshold is crossed and the consequence in case of violation. This pattern can be reused for any Metric in any “rule” or SLS in this representation rather than creating each time such a structure.

When it comes to the financial-related aspect and penalties associated to a consequence, we just do a pointer to the SLA contract.

b) Example of the JSON representation of an SLA Resource model


```
{
  "id": "http/www.acme.com/SlaManagement/sla/123444",
  "name": "HighSpeedDataSLA",
  "description": "SLA for high speed data.",
  "version": "0.1",
  "validityPeriod": {
          "startTime": "2013-08-01",
          "endTime": "2013-08-31"
      }
  "template": {
    "uri": "http/www.acme.com/slaManagement/slaTemplate/42",
    "name": "DataSLATemplate",
    "description": "basic template for Data SLA"
  },
  "involvedParties": [
    {
      "role": "SLAProvider",
      "reference": "http://"
```



```
    },
    {
      "role": "SLAConsumer",
      "reference": "http://"
    },
    {
      "role": "SLAAuditor",
      "uri": http://
    },

   {
      "role": "EndUser",
      "uri": "http://"
    }
],
“state:”: “observed”,
"approved": "true",
"approvalDate": "2013-09-10",

"rules": [
    {
      "id": "availability",
      "metric": "http://IEEE99.5/Availability ",
      "unit": "string",
      "referenceValue": "available",
      "operator": ".eq",
      "tolerance": "0.05",
      "consequence": "http://ww.acme.com/contract/clause/42",
    },
    {
      "id": "downstream_GBR",
      "metric": "http://IEEE99.5/Data/bitrates/GBR/down",
      "unit": "kbps",
      "referenceValue": "1024",
      "operator": ".ge",
      "tolerance": "0.20",
      "consequence": "http://ww.acme.com/contract/clause/45",
    },
]
```



```
}
```

SLAViolation Resource

a) SLA Violation resource model Description

This “SLAViolation” is the second Resource Model we are considering in this translation into JSON representation. So, we adopted the same representation we used for “SLA” resource.

We assume all KPIs (Metrics) are existing ones, stored in the Service Provider “Metrics Library” with their attached references.

We defined “Violations” in terms of ID, description (KPIs, reported date, period, start Time, end Time). Besides, the Involved Parties such as SLA Provider (CSP), SLA Consumer” (DSP) and SLA Auditor (for SLA monitoring) are represented in the same way we did for SLA” resource model. The same goes for “EndUser” role. The important thing we wanted to introduce in such an “Event” representation is to present an immediate view of the event result r impact. This practical and operational view allows the involved parties to react and perform an immediate and direct analysis of potential impacts of the violation.

We also introduced “Attachments” which represents statistics, a dashboard or reporting data to be presented to the target parties, DSP (SLA Consumer) or / and to CSP (SLA Provider) for deeper analysis purpose.

b) Example of the JSON representation of an SLA Violation

```
{
  "id": "http/www.acme.com/slaManagement/slaViolation/123444",
  "date": "2013-09-10 12:00:00Z",
  "sla": {
    "description": "sla of premium video"
    "uri": "http/www.acme.com/slaManagement/sla/123444",
```

```
},
"involvedParties": [
   {
      "role": "SLAProvider",
      "uri": "http://"
   },

   {
      "role": "SLAAuditor",
      "uri": http://
   },

   {
      "role": "SLAConsumer",
      "uri": "http://"
   },

  {
      "role": "EndUser",
      "uri": "http://"
   }
],
"violations": [{
   "rule": {
            "description": "availability",
            "uri": " http/www.zak.com/slaManagement/sla/123444/rules/availability",
    },

    "unit": "string",
    "referenceValue": "available",
    "operator": ".eq",
    "actualValue": "avalaible",
    "tolerance": "0.05",

   "violationAverage": "0.1",
   "comment": "Availability below agreed level.",
   "consequence": "http://ww.acme.com/contract/clause/42",

   "attachments": [{
      "description": "availability statistics for August 2013",
      "uri": "https://foo.bar/screenshot.123"
   }]
}]
```



```
}
```

Notification Resource Models

For every single event supported by the API provide a JSON based representation of the managed event.

You can start with an XML representation but remember that the default representation will be JSON.

Remember that the Pub/Sub models are common and described in the TMF REST Design Guidelines.

Following the available event types for SLA :

TMF REST	
SLAViolationEvent	body : SLAViolation

SLAViolation Event

a) SLA Violation Event Description

This event is identified through an Id and the occurrence date. It also refers to the associated SLA, the involved Parties and their attached roles and to the SLAViolation resource. Indeed the “event” as a notification alongside an operation is applied to a resource. In this respect, all this “SLAViolation” resource JSON representation we did in the previous section is reported in the current one. As in the case of “SLAViolation” resource, we represented the structure of the “SLA violation” in two steps from impact analyse perspective:

- a summarized view of the event result for an immediat analysis of potential impacts of the event;
- an “Attachment which contains statistics, dashboard for a deeper impact analyse.

In order to ease consequence handling by the OSS (Operations, escalation, automatic remedies), it is necessary to couple the alarm Id to the “network resource” Id in the “Network Resource Inventory” all stored in the CSP’s OSS. We assumed all KPIs (Metrics) are existing ones, stored in the Service Provider “Metrics Library” with their attached references and associated to products in the Catalogue.

So, this direct relationship of a given “SLAViolationEvent” to the violated Metric Id to the “network resource” in failed status and the Id of alarm raised allows an automated processing of impact analysis on the service. This interaction of the SLA Management to the CSP’s OSS in case of “SLAViolationEvent”, is described in TR 178V2.

b) Example of the JSON representation of an SLA Violation Event


```
{
  "event": {
 {
  "id": "http/www.acme.com/slaManagement/slaViolation/123444",
  "date": "2013-09-10 12:00:00Z",
  "sla": {
     "description": "sla of premium video"
     "uri": "http/www.acme.com/slaManagement/sla/123444",
  },
```



```
"involvedParties": [
   {
      "role": "SLAProvider",
      "uri": "http://"
   },
   {
      "role": "SLAAuditor",
      "uri": http://
   },
   {
      "role": "SLAConsumer",
      "uri": "http://"
   },
  {
      "role": "EndUser",
      "uri": "http://"
   }
],
"violations": [{
   "rule": {
            "description": "availability",
            "uri": "http/www.acme.com/slaManagement/sla/123444/rules/availability",
    },
    "unit": "string",
    "referenceValue": "available",
    "operator": ".eq",
    "actualValue": "avalaible",
    "tolerance": "0.05",
   "violationAverage": "0.1",
   "comment": "Availability below agreed level.",
   "consequence": "http://ww.acme.com/contract/clause/42",
   "attachments": [{
      "description": "availability statistics for August 2013",
      "uri": "https://foo.bar/screenshot.123"
   }]
}]
```



```
  },
   "eventType": "SLAViolation"
}
```


API OPERATION FOR SLA

For every single of operation on the entities use the following templates and provide sample REST requests and responses.

Remember that the following Uniform Contract rules must be used :

Operation on Entities	Uniform API Operation	Description
Query Entities	GET Resource	GET must be used to retrieve a representation of a resource.
Create Entity	POST Resource	POST must be used to create a new resource
Partial Update of an Entity	PATCH Resource	PATCH must be used to partially update a resource
Complete Update of an Entity	PUT Resource	PUT must be used to completely update a resource identified by its resource URI
Remove an Entity	DELETE Resource	DELETE must be used to remove a resource
Execute an Action on an Entity	POST on TASK Resource	POST must be used to execute Task Resources
Other Request Methods	POST on TASK Resource	GET and POST must not be used to tunnel other request methods.

Filtering and attribute selection rules are described in the TMF REST Design Guidelines.

Notifications are also described in a subsequent section.

GET /API/SLA/{ID}

This Uniform Contract operation is used to retrieve the representation of an agreement.

Note that collections can be retrieved via GET /API/SLA with no {ID}

Filtering is allowed on all attributes. See example below.
Attribute selection is possible for all attributes. See example below.

Template request for get: GET /api/sla/{ID }?{attributeSelector}&{filtering expression}

REQUEST
GET /api/sla Accept: application/json
RESPONSE
200 Content-Type: application/json {"sla": [}] }

Retrieving all SLAs – returns an array/ a list of SLAs:

- GET /api/sla

Retrieving all SLAs where ‘http://..//acme.com’ is involved with “SLAProvider” role:

- GET /api/sla? ….

Retrieving all SLAs where ‘http://…//mycompany.com’ is involved with “SLAConsumer” role:

- GET /api/sla? ….

Retrieving all SLAs where ‘http://…//John.Doe’ is involved with “EndUser” role:

- GET /api/sla? ….

Retrieving all SLAs based on template
“'http/www.acme.com/slaManagement/slaTemplate/42":

- GET /api/sla? ….

Retrieve sla with specified ID – only one SLA is returned:

- GET /api/sla/1

PUT API/SLA/{ID} (INTERNAL API)

Description:

- This Uniform Contract operation is used to completely update the representation of a SLA.
- Resource represents a managed entity.

Behavior:

- Returns HTTP/1.1 status code 201 if the request was successful.
- Returns HTTP/1.1 status code 400 (Bad request) if content is invalid (missing required attributes, …).

Updating the whole SLA – if you try to change the SLA ID itself an exception is returned. All fields with different values will be changed. If the request contains the same values like the current SLA representation, nothing is changed. If an element is empty in the request, the value of the element will be deleted. If it is a required element, an exception is returned.

REQUEST
PUT API/SLA/1 Content-type: application/json {"sla": { }}
RESPONSE

201
Content-Type: application/json
 	
{"sla": {
  }

Examples see TMF REST Design Guidelines.

PATCH API/SLA/{ID} (INTERNAL API)

Description:

- This Uniform Contract operation is used to partially update the representation of a SLA.
- Resource represents a managed entity.

REQUEST
PATCH API/sla/{ID} Content-type: application/json {"sla": { }}
RESPONSE
201 Content-Type: application/json {"sla": { }}

POST API/SLA (INTERNAL API)

Description:

- This Uniform Contract operation is used to create a SLA.
- Resource represents a managed entity.
- Mandatory attributes that must be provided when you create the SLA :

  Description, severity, type

Behavior:

- Returns HTTP/1.1 status code 201 if the request was successful.
- Returns HTTP/1.1 status code 400 (Bad request) if content is invalid (missing required attributes, …).

The requester cannot generate the id – the id to identify the REST resource is generated automatically in the back-end. The correlationId can be set from external but is not mandatory.

Create a SLA only with mandatory attributes:

REQUEST

POST API/sla
Content-type: application/json


```
{"sla": {
   "description": "Customer complaint over last invoice.",
   "severity": "Urgent",
   "type": "Bills, charges or payment",
}}
```


RESPONSE

201
Content-Type: application/json


```
{"sla": {


}}
```

API NOTIFICATION FOR SLA

For every single of operation on the entities use the following templates and provide sample REST notification POST calls.

It is assumed that the Pub/Sub uses the Register and UnRegister mechanisms described in the REST Guidelines reproduced below.

REGISTER LISTENER POST /HUB

Description :

Sets the communication endpoint address the service instance must use to deliver information about its health state, execution state, failures and metrics. Subsequent POST calls will be rejected by the service if it does not support multiple listeners. In this case DELETE /api/hub/{id} must be called before an endpoint can be created again.

Behavior :

Returns HTTP/1.1 status code 204 if the request was successful.

Returns HTTP/1.1 status code 409 if request is not successful.

REQUEST
POST /api/hub Accept: application/json {"callback": "http://in.listener.com"}
RESPONSE
201 Content-Type: application/json Location: /api/hub/42 {"id":"42","callback":"http://in.listener.com","query":null}

UNREGISTER LISTENER DELETE HUB/{ID}

Description :

Clears the communication endpoint address that was set by creating the Hub.

Behavior :

Returns HTTP/1.1 status code 204 if the request was successful.

Returns HTTP/1.1 status code 404 if the resource is not found.

REQUEST
DELETE /api/hub/{id} Accept: application/json
RESPONSE
204

PUBLISH {EVENTTYPE} POST /LISTENER

Description:

Provide the Event description

Behavior:

Returns HTTP/1.1 status code 201 if the service is able to set the configuration.

REQUEST

POST /client/listener
Accept: application/json
 	
{
 	
"event": {
 	
EVENT BODY
},


```
  "eventType": "eventType"
}
```


RESPONSE

201
Content-Type: application/json

Examples see TMF REST Design Guidelines.

API OPERATION FOR SLAVIOLATION

For every single of operation on the entities use the following templates and provide sample REST requests and responses.

Remember that the following Uniform Contract rules must be used :

Operation on Entities	Uniform API Operation	Description
Query Entities	GET Resource	GET must be used to retrieve a representation of a resource.
Create Entity	POST Resource	POST must be used to create a new resource
Partial Update of an Entity	PATCH Resource	PATCH must be used to partially update a resource
Complete Update of an Entity	PUT Resource	PUT must be used to completely update a resource identified by its resource URI
Remove an Entity	DELETE Resource	DELETE must be used to remove a resource
Execute an Action on an Entity	POST on TASK Resource	POST must be used to execute Task Resources
Other Request Methods	POST on TASK Resource	GET and POST must not be used to tunnel other request methods.

Filtering and attribute selection rules are described in the TMF REST Design Guidelines.

Notifications are also described in a subsequent section.

GET /API/SLAVIOLATION/{ID}

This Uniform Contract operation is used to retrieve the representation of an agreement.

Note that collections can be retrieved via GET /API/SLAVIOLATION with no {ID}

Filtering is allowed on all attributes. See example below.
Attribute selection is possible for all attributes. See example below.

Template request for get: GET /api/slaviolation/{ID}/{attributeSelector}?{filtering expression}

REQUEST
GET /api/slaviolation Accept: application/json
RESPONSE
200 Content-Type: application/json {"slaviolation": [}] }

Retrieving all SLAViolations – returns an array/ a list of SLAViolations:

- GET /api/slaviolation

Retrieving all SLAViolations where 'http://..//acme.com' is involved with "SLAProvider" role:

- GET /api/slaviolation? ….

Retrieving all SLAViolations where 'http://…//mycompany.com' is involved with "SLAConsumer" role:

- GET /api/slaviolation? ….

Retrieving all SLAViolations where ‘http://…//John.Doe’ is involved with “EndUser” role:

- GET /api/slaviolation? ….

Retrieving all SLAViolations referring to SLA " http/www.acme.com/SlaManagement/sla/123444”:

- GET /api/slaviolation? ….

Retrieving all SLAViolations where rule "http/www.acme.com/slaManagement/sla/123444/rules/availability” was violated

- GET /api/slaviolation? ….

Retrieving all SLAViolations where rule "http/www.acme.com/slaManagement/sla/123444/rules/availability” was violated between “12/11/2012” and “31/12/2013”.

- GET /api/slaviolation? ….

Retrieve slaviolation with specified ID – only one slaviolation is returned:

- GET /api/slaviolation/1

PUT API/SLAVIOLATION/{ID}

Not needed: a SLAViolation cannot be modified.

PATCH API/SLAVIOLATION/{ID}

Not needed: a SLAViolation cannot be modified.

POST API/SLAVIOLATION (INTERNAL API)

Description :

- This Uniform Contract operation is used to create a SLAViolation.
- Resource represents a managed entity.
- Mandatory attributes that must be provided when you create the SLAViolation

Behavior:

- Returns HTTP/1.1 status code 201 if the request was successful.
- Returns HTTP/1.1 status code 400 (Bad request) if content is invalid (missing required attributes, …).

The requester cannot generate the id – the id to identify the REST resource is generated automatically in the back-end.

Create a SLAViolation only with mandatory attributes:

REQUEST
POST API/slaviolation Content-type: application/json {"slaviolation": { }}
RESPONSE
201 Content-Type: application/json {"slaviolation": {   }}

API NOTIFICATION FOR SLAVIOLATION

For every single of operation on the entities use the following templates and provide sample REST notification POST calls.

It is assumed that the Pub/Sub uses the Register and UnRegister mechanisms described in the REST Guidelines reproduced below.

REGISTER LISTENER POST /HUB

Description:

Sets the communication endpoint address the service instance must use to deliver information about its health state, execution state, failures and metrics. Subsequent POST calls will be rejected by the service if it does not support multiple listeners. In this case DELETE /api/hub/{id} must be called before an endpoint can be created again.

Behavior :

Returns HTTP/1.1 status code 204 if the request was successful.

Returns HTTP/1.1 status code 409 if request is not successful.

REQUEST
POST /api/hub Accept: application/json {"callback": "http://in.listener.com"}
RESPONSE
201 Content-Type: application/json Location: /api/hub/42 {"id":"42","callback":"http://in.listener.com","query":null}

UNREGISTER LISTENER DELETE HUB/{ID}

Description:

Clears the communication endpoint address that was set by creating the Hub.

Behavior:

Returns HTTP/1.1 status code 204 if the request was successful.

Returns HTTP/1.1 status code 404 if the resource is not found.

REQUEST
DELETE /api/hub/{id} Accept: application/json
RESPONSE
204

PUBLISH {EVENTTYPE} POST /LISTENER

Description:

Provide the Event description

Behavior:

Returns HTTP/1.1 status code 201 if the service is able to set the configuration.

REQUEST

POST /client/listener
Accept: application/json
 	
{
 	
"event": {
 	
EVENT BODY
},


```
  "eventType": "eventType"
}
```


RESPONSE

201
Content-Type: application/json

Examples see TMF REST Design Guidelines.

APPENDIX A. : API SUMMARY

Entity	Operation	Part of Standard	Priority	Comments
SLA	*GET*	Y	1	required
SLA	*PUT*	N	2	Needed for Hackathon
SLA	*PATCH*	N	3	Future (negotiation)
SLA	*POST*	N	3	Future (negotiation)
SLAViolation	*GET*	Y	1	required
SLAViolation	*POST*	N	3	Needed for Hackathon
SLAViolation	*notification*	Y	1	

RELEASE HISTORY

Version Number	Date	Release led by:	Description
Version 0.5	11/09/2013	Jérôme Hannebelle Orange jerome.hannebelle@orange.com	First Release of Draft Version of the Document.
Version 0.6	Sept 2013	Tayeb BEN MERIEM	Improved the initial test drafted for describing each section: SLA, SLAvilolation and SLAviolation event.
Version 0.7	Sept 2013	Tina O'Sullivan (TM Forum)	Minor Updates & corrections.
V ersion 0.7.1	Oct 2013	Alicja Kawecki (TM Forum)	Corrections to cover, header & footer

ACKNOWLEDGEMENTS

Release Number	Date	Contributor	Company	Email
1.0	11th Sept 2013	Zavisa Bjelogrlic	Accenture	zavisa.bjelogrlic@accenture.com
		Tayeb BEN MERIEM	Orange	tayeb.benmeriem@orange.com
		Jerome Hannebelle	Orange	jerome.hannebelle@orange.com
		Pierre Gauthier	TM Forum	pgauthier@tmforum.org