TM Forum Technical Report

Metrics and Observations

TR292G

Maturity Level: **General availability (GA)**	Team Approved Date: 04-Jul-2024
Release Status: Production	Approval Status: TM Forum Approved
Version 3.6.0	IPR Mode: RAND

Notice

This document and the information contained herein is provided on an “AS IS” basis and TM FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

TM FORUM invites any TM FORUM Member or any other party that believes it has patent claims that would necessarily be infringed by implementations of this TM Forum Standards Final Deliverable, to notify the TM FORUM Team Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the TM FORUM Collaboration Project Team that produced this deliverable.

The TM FORUM invites any party to contact the TM FORUM Team Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this TM FORUM Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the TM FORUM Collaboration Project Team that produced this TM FORUM Standards Final Deliverable. TM FORUM may include such claims on its website but disclaims any obligation to do so.

TM FORUM takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this TM FORUM Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on TM FORUM's procedures with respect to rights in any document or deliverable produced by a TM FORUM Collaboration Project Team can be found on the TM FORUM website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this TM FORUM Standards Final Deliverable, can be obtained from the TM FORUM Team Administrator. TM FORUM makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

Direct inquiries to the TM Forum office:

181 New Road, Suite 304
Parsippany, NJ 07054, USA
Tel No. +1 862 227 1648
TM Forum Web Page: www.tmforum.org

Table of Contents

Executive Summary

This document defines an ontology model for defining and using metrics and managing observations, such as measurements concerning the metrics.

Introduction

Intent requirements are often based on setting target values for measurable properties such as a KPI value or a state being reached. For example, conditions contributing to intent expectations or context depend on them. These are in general metrics and observations about them. It is possible in intent models to use URIs that directly identify a node in the knowledge graph, where the value to be used is allocated and kept up to date. But this would mean that intent owners need the exact URI the measurement implementation of the intent handler uses for a particular measurement. This requires that intent handlers expose many details, and it also is limited regarding contextual information about the metric and its observation. For example, the time of latest value update or the scope of the value presented. An example of measurement scope is a sunset of resources or users included in the measurement. For example a measurement of latency shall be the average over a defined set of users, but excluding other users. Managing these use cases requires a foundational model regarding metrics and dedicated vocabulary in the intent ontology.

Revision Information

This revision v3.6.0 of Metrics and Observations is part of the TM Forum Intent Ontology (TIO) v3.6.0.

The revision v3.6.0 of this document replaces v.3.5.0 with the following changes:

- Minor editorial corrections.

1. Notation and Namespaces

The metrics and observations model is part of the TM Forum Intent Ontology and depends on the following models:

Model	Prefix	Namespace	Published by	Purpose in the model
Metrics and observation	met	`http://tio.models.tmforum.org/tio/v3.6.0/MetricsAndObservations/`	TM Forum	Identifies the model defined in this document The metrics and observation model introduces metrics as measurable and observable items and defines vocabulary to manage observations about the metric
Set Operators Ontology	set	`http://tio.models.tmforum.org/tio/v3.6.0/SetOperators/`	TM Forum	Specification of set operators (This document)
Function Definition Ontology	fun	`http://tio.models.tmforum.org/tio/v3.6.0/FunctionOntology/`	TM Forum	Basic expression of functions
Quantity Ontology	quan	`http://tio.labs.tmforum.org/tio/v3.6.0/QuantityOntology`	TM Forum	Introduction of quantities and related operators
RDF version 1.1	rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	W3C	Providing fundamental modeling artifacts
RDF Schema 1.1	rdfs	`http://www.w3.org/2000/01/rdf-schema#`	W3C	Providing fundamental modeling artifacts
XML Schema	xsd	`http://www.w3.org/2001/XMLSchema#`	W3C	Providing of data types for literal objects
Time Ontology in OWL	t	`http://www.w3.org/2006/time#`	W3C	Expression of date and time [owltime]
Examples	ex:	`http://www..example.org/`	IANA	Reserved domain name for examples

Table 1.1: Model references

The metrics and observations model is based on the Resource Description Framework (RDF) [rdf, rdf_mt, rdf_primer] and the Resource Description Framework Schema (RDFS) [rdfs] published by the World Wide Web Consortium (W3C).

2. Metrics

A metric is an observable and measurable item. Examples are Key Performance Indicators (KPI) or states of resources and software. A metric has therefore a value that can be observed or measured within the operated infrastructure.

A metric is represented in the TM Forum Intent Ontology property. It is identified by a sub-property of `met:metric`.

The property `tio:associatedValueType` defines the type of values that would be observed for this metric.

For example:

```
ex:numberOfUsers
  a met:metric ;
  tio:associatedValueType xsd:integer
.
ex:sliceLatency
  a met:metric ;
  tio:associatedValueType quan:quantity
.
ex:serviceError
  a met:metric ;
  tio:associatedValueType xsd:string
.
ex:serviceState
  a met:metric ;
  tio:associatedValueType ex:ServiceAvailability
.
ex:Up
  a ex:ServiceAvailability
.
ex:Down
  a ex:ServiceAvailability
.
ex:ServiceLocations
  a met:metric ;
  tio:associatedValueType rdfs:Container
.
```

This example demonstrates multiple metric definitions with various metric value types. The value of a metric can be a literal with its type based on XML schema data types. In this example, the value type of the metric `ex:NumberOfUsers` is integer.

The value of a metric can also be an individual of any class. This example specifies that the value type for the metric `ex:sliceLatency` would be a quantity, thus, it consists of a numerical value in combination with a unit.

The value can hold a string. This is shown for the metric `ex:serviceError`. Its value would contain an error statement received about a service.

A state of a service or resource often has distinct values. An example of how this could be represented is the metric `ex:serviceState`. Its value would be an individual of class `ex:ServiceAvailability`. The example further introduces two individuals of this class `ex:Up` and `ex:Down`. These are the possible values for the `ex:serviceState` metric.

The value of a metric might also be a set of individual objects. A possibility to resent this would be by using a container as value of the metric. In the example this is shown for the metric ex:ServiceLocations. Members of this value container would be geographical locations or areas.

3. Observations

An observation is a distinct value or fact that was observed or measured for a metric. An observation is represented by an individual of class `met:Observation`. They represent a measured or otherwise observed fact about a resource. Examples are KPI values, aggregated insights or observed resource states.

The property `met:observedMetric` states what was observed by an observation. This refers to a `met:metric` property that identifies a metric.

The property `rdf:value` states the observed value obtained for a metric.

The property `met:observed` can be used to assign an instance of class `met:Observation` to other objects.

Intent extension models can define further properties to be used with an observation object to cover the different needs of metrics or KPIs.

For example:

```
ex:OB1
  a met:Observation
  met:observedMetric ex:numberOfUsers ;
  rdf:value "1254"^^xsd:integer
.
ex:OB2
  a met:Observation
  met:observedMetric ex:sliceLatency ;
  rdf:value [ quan:value "10.5"^^xsd:decimal ;
              quan:unit "ms" ]
.
ex:OB3
  a met:Observation
  met:observedMetric ex:serviceError ;
  rdf:value "No Error"^^xsd:string
.
ex:OB4
  a met:Observation
  met:observedMetric ex:serviceState ;
  rdf:value ex:Up
.
ex:OB5
  a met:Observation ;
  a met:observedMetric ex:serviceLocations ;
  rdf:value [ a rdfs:Container ;
              rdfs:member ex:Stockholm, Gothenburg, ex:Uppsala
            ]
.
```

This example shows observations for the metrics introduced in the example in chapter 2.

The observation `ex:OB1` states that 1254 users were observed.

The observation `ex:OB2` provides an observed value for the slice latency metric. The observed value is a quantity with a numerical value of 10.5 and the unit "`ms`".

The observation `ex:OB3` contains the error code of a service. In this example the string "`No error`" was obtained from the service indicating normal operation without issues.

The observation `ex:OB4` states the state of a service using pre-defined state individuals of class `ex:ServiceAvailability`. In this example the state shows that the service is up indicating that it is available for users to use.

The observation `ex:OB5` contains the locations the service is currently available at. The observed value is a set of three locations.

3.1. Observation time

An observation is obtained at a particular point in time. The property `met:obtainedAt` states the point in time when the observation was obtained and the observation individual was created in the knowledge base.

Due to latencies in the measurement infrastructure it takes some time from a value being valid in the observed system until the value is provided to the knowledge base, where it would be represented by observation individuals. The timestamp provided by `met:obtainedAt` represents when the observed value reached the knowledge base. Further timestamps, which for example represents respective value changes in the infrastructure more accurately are not in scope of this model, but can be introduced through intent extension models if needed.

For example:

```
ex:OB1
  a met:Observation
  met:observedMetric ex:numberOfUsers ;
  rdf:value "1254"^^xsd:integer ;
  met:obtainedAt [ t:inXSDDateTimeStamp "2023-07-
05T01:00+01:00"^^xsd:dateTimeStamp ]
.
```

This example shows an individual observation for the metric ex:NumberOfUsers which was obtained in the knowledge base on the 5th of July 2023 at 01:00 at night in the time zone GMT+1.

4. Observation Functions

Observation functions can be used to refer to and select individual observations for a metric:

`met:lastObserved`

`<met:Observation> = met:lastObserved ( <met:metric> ... )`

The function `met:lastObserved` refers to the newest individual observation created for the metrics that are provided as function arguments. For example, it refers to latest measurement obtained for a latency.

`met:observedValue`

`<value of observation> = met:observedValue ( <met:Observation> )`

The function `met:observedValue` refers to the value of an individual observation and provides this value as function result.

`met:lastValue`

`<value of observation> = met:lastValue ( <met:metric> ... )`

The function `met:lastValue` refers to the newest observed value for a metric and provides the value as result.

For example:

```
ex:Intent1
  a icm:Intent ;
  log:allOf ( ex:E1 ex:E2 )
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1 ;
  log:allOf ( ex:C1 )
.
ex:C1
  a icm:Condition ;
  quan:smaller ( [ met:lastValue ( ex:sliceLatency ) ]
                 [ rdf:value "10"^^xsd:decimal ;
                   quan:unit "ms" ]
               )
.
```

This example shows an intent definition with a property expectation depending on a condition. The condition states that the last observed value of a slice latency shall be smaller than 10ms.

5. Administrative Appendix

5.1. Document History

5.1.1. Version History

Version Number	Date Modified	Modified by:	Description of changes
3.4.0	29-Feb-2024	Alan Pope	Final edits prior to publication
3.5.0	03-May-2024	Alan Pope	Final edits prior to publication
3.6.0	04-Jul-2024	Alan Pope	Final edits prior to publication

5.1.2. Release History

Release Status	Date Modified	Modified by:	Description of changes
Pre-production	29-Feb-2024	Alan Pope	Initial release
Production	26-Apr-2024	Adrienne Walcott	Updated to reflect TM Forum Approved status
Pre-production	03-May-2024	Alan Pope	Updated to v3.5.0
Production	28-Jun-2024	Adrienne Walcott	Updated to reflect TM Forum Approved status
Pre-production	04-Jul-2024	Alan Pope	Updated to v3.6.0
Production	30-Aug-2024	Adrienne Walcott	Updated to reflect TM Forum Approved status

5.2. Acknowledgments

Team Member (@mention)	Company	Role*
Jörg Niemöller	Ericsson	Author, Project Co-Chair
Kevin McDonnell	Huawei	Project Co-Chair
Yuval Stein	Amdocs	Project Co-Chair
Kamal Maghsoudlou	Ericsson	Key Contributor
Leonid Mokrushin	Ericsson	Key Contributor
Marin Orlić	Ercisson	Key Contributor
Aaron Boasman-Patel	TM Forum	Additional Input
Alan Pope	TM Forum	Additional Input
Dave Milham	TM Forum	Additional Input
Xiao Hongmei	Inspur	Reviewer

**Select from: Project Chair, Project Co-Chair, Author, Editor, Key Contributor, Additional Input, Reviewer*

6. TR292G Metrics and observations v3.6.0 - Appendix A Vocabulary

6.1. Appendix A: Vocabulary Reference

This Appendix chapter contains a reference definition of all model vocabulary. It is sorted alphabetically:

6.1.1. lastObserved

The function `met:lastObserved` refers to the newest individual observation created for the metrics that are provided as function arguments. For example, it refers to latest measurement obtained for a latency.

Instance of: `fun:function`Result type: `met:Observation`Arity: at least one
Argument types: All arguments `are rdfs:Metric`

6.1.2. lastValue

The function `met:lastValue` refers to the newest observed value for a metric and provides the value as result.

Instance of: `fun:function`Result type: `rdf:Resource`Arity: at least one
Argument types: All arguments `are rdfs:Metric`

6.1.3. metric

Sub-properties of `met:Metric` represent observable and measurable items.

Instance of: `rdf:Property`

6.1.4. Observation

An observation is a distinct value or fact that was observed or measured for a metric. An observation is represented by an individual of class `met:Observation`. They represent a measured or otherwise observed fact about a resource.

Instance of: `rdfs:Class`

6.1.5. observed

The property `met:observed` can be used to assign an instance of class `met:Observation` to other objects.

Instance of: `rdf:Property`Range: `met:Observation`

6.1.6. observedMetric

The property `met:observedMetric` states what was observed by an observation. This refers to a `met:metric` property that identifies a metric.

Instance of: `rdf:Property`Domain: `met:Observation`Range: `met:Metric`

6.1.7. observedValue

The function `met:observedValue` refers to the value of an individual observation and provides this value as function result.

Instance of: `fun:function`Result type: `rdf:Resource`Arity: exactly 1
Argument types: `met:Observation`

6.1.8. obtainedAt

The property `met:obtainedAt` states the point in time when the observation was obtained and the observation individual was created in the knowledge base.

Instance of: `rdf:Property`Domain: `met:Observation`

6.1.9. Vocabulary

The object `met:Vocabulary` is a container of all model elements.

Instance of: `rdfs:Container`