TM Forum Technical Report

Intent Common Model - Intent Expression

TR290A

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

©TM Forum 2024. All Rights Reserved.

Notice

Copyright © TM Forum 2024. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to TM FORUM, except as needed for the purpose of developing any document or deliverable produced by a TM FORUM Collaboration Project Team (in which case the rules applicable to copyrights, as set forth in the TM FORUM IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by TM FORUM or its successors or assigns.

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 is part of the TR290 series specifying the intent common model as part of the TM Forum Intent Ontology. The focus of this document is the specification of how intent is structured and expressed.

Introduction

The TM Forum Intent Ontology (TIO) is a model of intent expression and management. A major part of the model is the intent common model specified in the TR290x series of documents. It is mandatory for an intent management function to support at least one version of the intent common model. It contains generic concepts of intent models and expression vocabulary. This means its vocabulary is not specific to an application domain or particular use case. Rather, it defines concepts that are broadly useful for intent expression in any domain and intent use case. The use case specific and optional models are referred to as intent extension models and are not in the scope of the TR290x series of documents.

Scope

This document is part of the intent common model. It specifically covers the ontology and vocabulary for expressing intent.

Revision Information

This revision v3.6.0 of the intent common model is part of the TM Forum Intent Ontology (TIO) v3.6.0. Changes are logged in the main document TR290.

1. Notation and Dependencies

The intent common model depends on the following models:

Model	Prefix	Namespaces	Published by	Purpose in the model
Intent Common Model	icm	`http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/`	TM Forum	General ontology model of intent and intent report expression. This document is part of the intent common model specification.
Intent Management Ontology	imo	`http://tio.models.tmforum.org/tio/v3.6.0/IntentManagmentOntology/`	TM Forum	Defines basic vocabulary and concepts of intent based operation. This document specifies vocabulary for intent management functions and their roles, as well as the types of intent models within the TM Forum Intent Ontology (TIO).
Conditions and logical operators Ontology	log	`http://tio.models.tmforum.org/tio/v3.6.0/LogicalOperators/`	TM Forum	Specifies logical operators to express logical relationships and the evaluation of truth values.
Quantity Ontology	quan	`http://tio.models.tmforum.org/tio/v3.6.0/QuantityOntology/`	TM Forum	Introduces quantities and quantity operators.
Set Operators	set	`http://tio.models.tmforum.org/tio/v3.6.0/SetOperators/`	TM Forum	Specification of set operators.
Function Definition Ontology	fun	`http://tio.models.tmforum.org/tio/v3.6.0/FunctionOntology/`	TM Forum	Basic expression of functions.
Time Ontology in OWL	t	`http://www.w3.org/2006/time#`	W3C	Expression of date and time [owltime]
RDF version 1.1	`rdf`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	W3C	Providing fundamental modeling basics [rdf11]
RDF Schema 1.1	`rdfs`	`http://www.w3.org/2000/01/rdf-schema#`	W3C	Providing schema for knowledge modeling [rdfs11]
XML Schema	`xsd`	`http://www.w3.org/2001/XMLSchema#`	W3C	Providing data types for literal objects [xsd-1] [xsd-2]

© TM Forum 2024. All Rights Reserved.

Model	Prefix	Namespaces	Published by	Purpose in the model
Examples	ex	http://www..example.org/	IANA	Reserved domain name for examples

Table 1: Model references

The intent common 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).

Furthermore, the intent common model depends on base models of the TM Forum intent ontology. These are the intent management ontology, the function definition ontology, the logical operators ontology and the quantity ontology.

© TM Forum 2024. All Rights Reserved.

2. The structure of Intent

Intent is an object within the TM Forum Intent Ontology (TIO) that represents a set of requirements. It is a set of definitions and sub-graphs in the intent manager's knowledge base. Intent individuals are instances of class `icm:Intent`.

The intent common model specifies a structure of intent graphs as shown in Figure 1.

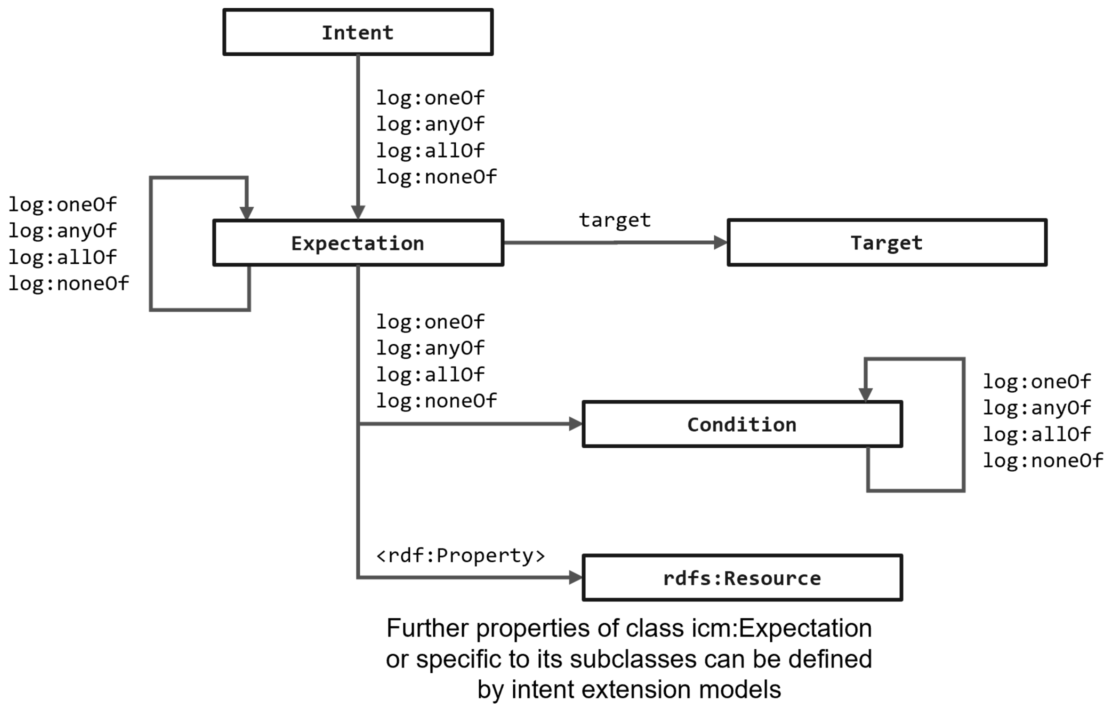

Figure 1: Intent structure

Intent contains a set of individuals of type `icm:Expectation`. Expectations are sub-sets of requirements, which have a common target. A target defines a set of resources needed to fulfill the requirements given by the expectations it is associated with, using the property `icm:target`. Expectations define details of the requirements through their properties, including conditions. The details of what types of requirements can be defined, and how to express them, depends on the type of expectation. Therefore, different sets of properties are defined for each type of expectation. One way to define required details within an expectation is condition objects. These express the exact condition a system has to meet to be considered compliant.

3. Evaluation of Compliance

3.1. Overall Compliance to the Intent

Intent has an associated truth value that is interpreted as compliance. It is true if the managed system is compliant to all requirements. The resulting overall compliance of the intent depends on the compliance contributions of its expectations. Expectations also have an associated truth value interpreted as compliance. It represents the compliance of the system to the specific subset of requirements defined by an expectation. Logical operators would be used to specify the logical evaluation of the intent compliance from expectation compliance. The functions defined in the logical operators model can be used to express the relationship and evaluation logic.

For example:

```
ex:Intent1
  a icm:Intent ;
  log:allOf ( ex:E1 ex:E2 )
.
```

This example considers that the intent defines two expectation objects `ex:E1` and `ex:E2`. It uses the logical operator function `log:allOf` to express logical conjunction. Consequently, the targeted resources are considered to be compliant to the intent overall, if it is individually compliant to the subsets of requirements defined by both expectations. Each expectation has an individual truth value derived from the evaluation of its requirements.

If multiple properties are given that provide a truth value, the overall compliance is evaluated considering logical conjunction.

For example:

```
ex:Intent1
  a icm:Intent ;
  log:allOf ( ex:E1 ex:E2 ) ;
  log:anyOf ( ex:E3 ex:E4 )
.
```

This example uses two more distinct expectation objects `ex:E3` and `ex:E4`. Their combined contribution to the evaluation of compliance to the intent is determined by a logical disjunction expressed by the `log:anyOf` operator. This contribution is evaluated in conjunction with the contribution from `ex:E1` and `ex:E2`. Overall, the targeted resources are considered to be compliant to the intent if they are compliant to `ex:E1` and `ex:E2` and either of the expectations `ex:E3` and `ex:E4`.

3.2. Compliance to Expectations

The truth value associated with an expectation is interpreted as compliance similar to the truth value associated with an intent. The value is derived from the properties assigned to an expectation individual. The definition of a property determines the details and how the property is evaluated individually to conclude a truth value.

Conditions can be used to define the compliance of expectations by defining an exact condition statement. A condition object has an associated truth value as well. It is derived from the properties of the condition object. Typically, functions providing a boolean result are used and combined to express the evaluation of the truth value. However, conditions are a universal concept. They can be used in other contexts where an evaluation is needed to determine a truth value and not only for compliance to an intent or expectation. It becomes a condition of compliance only if it is assigned to an intent or expectation.

For example:

```
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1 ;
  log:allOf ( ex:C1 )
.
ex:C1
  a log:Condition ;
  quan:smaller ( ex:Latency1
                 [ rdf:value "10"^^xsd:decimal ;
                   unit "ms" ]
               )
.
```

This example defines a condition object `ex:C1` that gets its truth value from the evaluation of a quan:smaller function. This function compares two quantities. The first one `ex:Latency1` represents the quantity of a latency KPI. For this example it is assumed that this individual exists in the knowledge base and its value would be updated by the KPI measurement infrastructure. Its value is compared to a constant expressing a given quantity of 10 milliseconds. Therefore, the condition is true, and the value associated with `ex:C1` is "true" while the measured quantity of `ex:Latency1` is smaller than 10 milliseconds. This result is associated to the truth value of the expectation `ex:E1` through the logical operator function `log:allOf`. This means the condition result is interpreted as a requirement that determines compliance.

The example above has used an expectation of type property expectation and a condition to state a requirement. Other types of expectations might use different properties explained in Chapter 6 and its subchapters that discuss various types of expectations.

4. Targets

When specifying a requirement, it is essential to not only express the requirement itself with respect to the exact conditions for compliance, but to specify the resources which are expected to comply to the requirement.

This can for example be a service that needs to be delivered to a customer. The service would be required to provide access to a video streaming platform and needs to comply to 99% availability. The service instance provisioned for this request is the resource that has to meet the requirement. Another example would be a network slice expected to provide a latency below the required threshold.

A target within an intent is an individual object of class `icm:Target`. It is defined as subclass of `rdfs:Container`. As such a target object would represent a set of resources. The members of the target container represent the resources that need to comply to a requirement. Which requirements depend on the assignment of the target object to expectation objects.

A target is assigned to an expectation using the `icm:target` property. Consequently, all requirements this expectation defines need to be met by the resources which are members of the target.

There is no inherent restriction on what the target can be. The intent common model intentionally defines targets very generically to allow for a great variety of possible targets as needed by a great variety of domains and use cases. The only requirement is that the target needs to be referable by an IRI/URI to be used within an RDF based intent model. For environments without global IRI/URI references, it is possible to create connectors representing proxy instances. The TM Forum Intent Ontology defines connector models [TR293] for this purpose.

For example:

```
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice1
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1 ;
  log:allOf ( ex:C1 )
.
ex:C1
  a log:Condition ;
  set:forAll ( _:X
             [ icm:valuesOfTargetProperty ( ex:latency1 ) ]
             quan:smaller ( _:X [ rdf:value "10"^^xsd:decimal ;
                                  unit "ms"
                                ] )
           )
.
```

This example defines the target `ex:T1` and specifies that the network slice individual `ex:Slice1` is a member of the target container. The target `ex:T1` is assigned to the expectation `ex:E1`. This means that `ex:Slice1` is expected to meet the requirements defined by `ex:E1`. Here the requirement is specified by a condition. Overall the example expresses that the slice `ex:Slice1` needs to show a latency below 10 milliseconds. The function `icm:valuesOfTargetProperty` generates the collection of values of property `ex:latency1` for all members of the target collection in the current scope of the condition `ex:C1`. The universal quantifier `set:forAll` iterates over that collection by binding variable `_:X` to every element of it, and verifies that the inequality specified by the function `quan:smaller` holds for all elements of the collection.

Intent can contain multiple expectations with different targets. This means that a single intent can combine many requirements not only with respect to requirement type, but also covering multiple distinct targeted resources with individual subsets of requirements.

Multiple expectations can have the same target. This means that the overall set of requirements the targeted resource must comply to is distributed over multiple distinct expectations.

If an expectation has multiple targets, the same expectation is used to define requirements for multiple sets of resources.

4.1. Individual Targets

Requirements associated with this target have to be fulfilled by all resources individually. If any of the targeted resources does not meet the requirement, the evaluation of the associated expectations would indicate that the target is not compliant, even if some target resources comply. For evaluating the overall compliance status of an expectation, the permutation of members of the target container and expectation requirements need to be checked and aggregated with a logical conjunction of the partial results.

For example:

```
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice1 ,
              ex:Slice2
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1 ;
  log:allOf ( ex:C1 )
.
ex:C1
  a log:Condition ;
  set:forAll ( _:X
               [ icm:valuesOfTargetProperty ( ex:latency1 ) ]
               quan:smaller ( _:X [ rdf:value "10"^^xsd:decimal ;
                                    unit "ms"
                                  ] )
             )
.
```

This example shows the target `ex:T1` being defined by enumerating all resource individuals. It contains two slice individuals `ex:Slice1` and `ex:Slice2`. The target is then assigned to the expectation `ex:E1`, which uses the condition `ex:C1` as a requirement. The condition states that the observed latency shall be below 10 milliseconds. The intent handler would individually check the condition for `ex:Slice1` and `ex:Slice2` using their respective observations of `ex:latency1` property. Assuming that the measurement of `ex:latency1` for `ex:Slice1` is 11 milliseconds and for `ex:Slice2` it is 9 milliseconds, the target only partially complies to the requirement. Consequently, the overall compliance result for expectation `ex:E1` would indicate that as a whole it is not compliant.

The assignment of members to a target container can be dynamic using abstract functions for container construction based on criteria and filters. This means that also resource instances, which are newly created, would be considered part of the target if and while they meet the target selection criteria. This is useful for requirements that shall be generally valid for a range of individual resources in an environment where these individuals are dynamically created and deleted. It removes the need to explicitly keeping the target members synchronized with the resources actually present at any point in time.

For example:

```
ex:T1
  a icm:Target ;
  set:intersection ( set:resourcesOfType ( ex:VideoService )
                     set:resourcesWithPropertyObject ( ex:ServiceLevel
ex:GoldService )
                   )
.
```

This example defines target `ex:T2` using container combination and filtering functions from the set operators ontology. These functions determine which of the resources in the knowledge base should be considered to be a member of the target. In this example two containers of resources are defined. The first shall contain all resources that are an instance of class `ex:VideoService`. The second container shall be resources that have the service level property indicating a gold level service. The members of the target are the intersection of both sets. This refers to all video service instances with gold service level.

4.2. Implied observation context

The expression of requirements is typically based on an observable property of a resource. For example a slice latency is an observable property of a network slice instance. There usually exists a measurement infrastructure that regularly measures a new value of this property and makes it available to management functions such as intent managers. If there are multiple instances of a resource such as a network slice, there would be multiple distinct sets of measurements. This means that the observation is typically contextualized per individual managed resource. Consequently, the evaluation of requirements need to be contextualized as well.

We need to distinguish two fundamental interpretations of a requirement and its context: The requirement might be applicable to each resource from the target container individually or to the entire set of resources in the target container together. Which interpretation is correct depends on the nature and definition of the used metric. It is interpreted as individual requirement if the used metric is a property of the resource type of individual element of the target. Respectively it is interpreted as

© TM Forum 2024. All Rights Reserved.

requirement of the combined set of resources, if the used metric is defined accordingly. The latter would be applicable for an aggregate metric.

For example

```
ex:T1
  a icm:Target ;
  rdfs:member ex:RBS1, ex:RBS2
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1 ;
  log:allOf ( ex:C1 ex:C2 )
.
ex:C1
  a log:Condition ;
  quan:smaller (
    [ quan:sum ( [ icm:valuesOfTargetProperty ( ex:powerUsage ) ] ) ]
    [ rdf:value 10.0 ;
      unit "kW"
    ]
  )
.
ex:C2
  a log:Condition ;
  set:forAll ( _:X
               [ icm:valuesOfTargetProperty ( ex:powerUsage ) ]
               quan:smaller ( _:X [ rdf:value 1"^^xsd:decimal ;
                                    unit "kW"
                                  ] )
             )
.
```

This example uses a target that defines two individual radio base stations. The target is assigned to expectation `ex:E1`, which uses conditions `ex:C1` and `ex:C2`. Both conditions must be fulfilled. The conditions define requirements regarding power usage. It is important for the evaluation if the conditions refer to individual power usage of every resource in the target or combined power usage of all resources in the target container together.

It is also possible to express that the requirement shall be applied to various subsets of the target container. If this is needed, multiple targets can be defined representing each distinct subset to evaluate.

For Example:

```
ex:T1
  a icm:Target ;
  rdfs:member ex:RBS1, ex:RBS2
.
ex:T2
  a icm:Target ;
  rdfs:member ex:RBS3, ex:RBS4
.
ex:E1
```

```
  a icm:PropertyExpectation ;
  icm:target ex:T1 ;
  log:allOf ( ex:C1 )
.
ex:E2
  a icm:PropertyExpectation ;
  icm:target ex:T2 ;
  log:allOf ( ex:C1 )
.
ex:C1
  a log:Condition ;
  quan:smaller (
    [ quan:sum ( [ icm:valuesOfTargetProperty ( ex:powerUsage ) ] ) ]
    [ rdf:value 10.0 ;
      unit "kW"
    ]
  )
.
ex:E3
  a icm:PropertyExpectation ;
  icm:target [ set:union ( ex:T1 ex:T2 ) ] ;
  log:allOf ( ex:C2 )
.
ex:C2
  a log:Condition ;
  set:forAll ( _:X
               [ icm:valuesOfTargetProperty ( ex:powerUsage ) ]
               quan:smaller ( _:X [ rdf:value 1"^^xsd:decimal ;
                                    unit "kW"
                                  ] )
             )
.
```

This example defines targets involving four distinct Radio Base Stations, split between two target containers. Both targets are associated with the expectation ex:E1 and two conditions. Condition ex:C1 is based on an aggregate power usage metric. It is evaluated two times with each set of RBS resources. The condition ex:C2 is based on an individual metric per RBS. It is evaluated four times individually for each of the four RBS within the targets.

4.3. Target Selection

An intent handler can select resources most suitable to fulfil the requirements. This would allow the intent handler to optimize operation. However, a pre-determined target does not allow any freedom to make a selection, because it prescribes the exact resources the requirements within the intent shall be applicable for. A dynamic choice of target members is steered by the `icm:chooseFrom` property of a delivery expectation. This is described in Chapter 6.3.

Targets might be specified in an intent without explicitly assigned members. The intent handler is supposed to choose the members most suitable to reach compliance to the requirements and by optimizing various concerns. It is however possible that no resources are selected, and the target stays an empty container. An expectation with an empty or missing target defaults to not compliant.

It is important to note that the assignment and selection of targets is dynamic and primarily depends on the resources within the graph of the knowledge base at any given point in time. There is not a particular time or deadline for explicitly evaluating the target criteria and create a container. Its members are rather the logical result of the knowledge present. This means that a change in the knowledge graph would have an immediate effect on the assumed members of a target, and by that on the evaluation of compliance.

4.4. Target Functions

The functions defined in this chapter allow dealing with targets and their member elements.

`icm:valuesOfTargetProperty`

`<rdfs:Container> = icm:valuesOfTargetProperties ( <rdf:Property> ... )`

The function `set:valuesOfTargetProperty` represents a container of all values that are assigned to members of the target container with the properties specified as function argument. If multiple properties are specified as arguments the result is a superset of all values from all specified properties.

For example:

```
ex:T1
  a icm:Target ;
  icm:chooseFrom ( [ set:resourcesOfType ( ex:PremiumSlice ) ] )
.
ex:T2
  a icm:Target ;
  icm:chooseFrom ( [ set:resourcesOfType ( ex:Slice ) ] )
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1 ;
  log:allOf ( ex:C1 ex:C2 )
.
ex:E2
  a icm:PropertyExpectation ;
  icm:target ex:T2 ;
  log:allOf ( ex:C1 )
.
ex:C1
  a log:Condition ;
  quan:smaller ( quan:mean ( icm:valuesOfTargetProperties ( ex:latency ) )
               "10ms"^^quan:quantity
             )
.
```

This example uses the function `icm:valuesOfTargetProperties` to represent a container of all latency values of all slices that are members of the target. This is part of a condition, which is then used by expectations. The condition is that the mean of all latency values associated with slices chosen in the target shall be below `10ms`. The target changes depending on the expectation. This means that the condition evaluated as part of expectation `ex:E1` is evaluated based on the respective target `ex:T1` and its members. The same condition evaluated for expectation `ex:E2` would be based on target `ex:T2` and its members. This means that condition can provide different results for both expectations. The function `icm:valueOfTargetProperties` is target context aware and the diffrentiation depending on expectation and target is based on this function.

© TM Forum 2024. All Rights Reserved.

5. Expectations

Expectation objects represent a set of requirements within the expression of an intent. The intent common model defines the class of `icm:Expectation` and multiple subclasses of it. Each subclass represents different type of requirements allowing the expression of different concerns. For example, describing the functional properties of a required service is a different concern than expressing its expected performance. The expression of a functional requirement would be based on parameters that describe and categorize needed functional aspects. Performance requirements are typically expressed using measured metrics and KPIs. These different expression concerns translate into different types of expectations.

Expectations have targets. This means that they associate a set of requirements with the resources that are expected to comply to and fulfill these requirements. Targets are assigned with the property `icm:target`. Its domain is the class `icm:Expectation` and its range is the class `icm:Target`.

A Boolean truth value is associated with an expectation. It is interpreted as an indication of compliance of the target resources of the expectation instance to the requirements specified by the expectation. The main purpose of the properties of an expectation is therefore to specify the details for determining its associated compliance truth value.

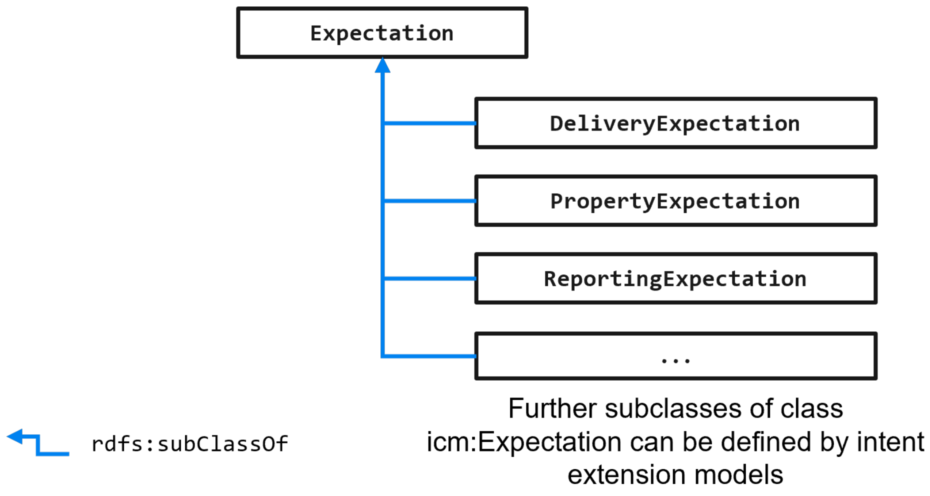

Figure 2: Subclasses of `icm:Expectation`

Next to the types of expectations defined in the intent common model, extension models are allowed and encouraged to introduce new subclasses of `icm:Expectation` if the expressiveness of the common classes is not sufficient for their use cases.

6. Delivery Expectation

The expectation class `icm:DeliveryExpectation` is used for requiring that something needs to be delivered, and it allows specifying what this something is. This is practically a functional requirement. For example, a delivery expectation might require a service to be delivered, and it would allow specifying what type of service is needed and by whom. The target is in this respect the resource that is delivered or manages the delivery.

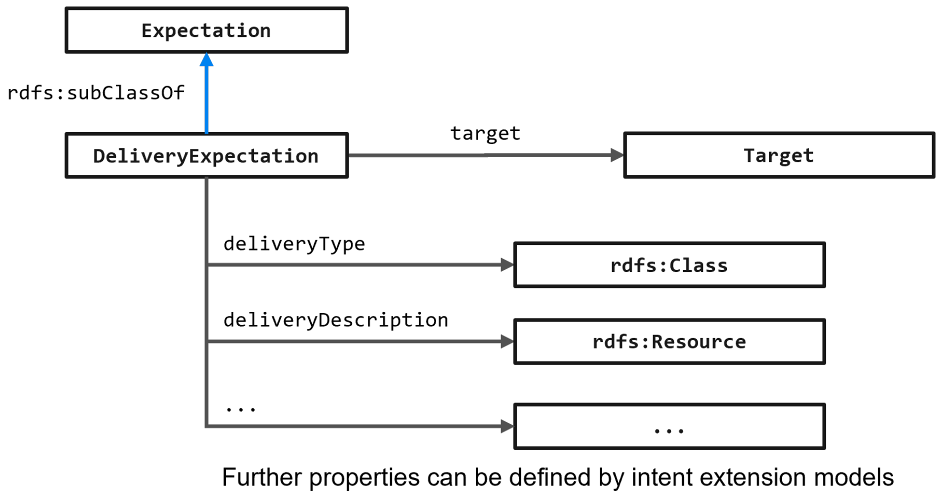

Figure 3: Delivery Expectation

The intent common model defines two properties for describing what needs to be delivered. Intent extension models can introduce further properties and extended vocabulary needed for particular domains and their use cases.

6.1. Type of resource to be deliverd: icm:deliveryType

The property `icm:deliveryType` requires that the delivered resource shall be an instance of the specified class. A target resource is compliant if its type matches.

The delivery of a resource involves that a resource is chosen from a defined set of options as described in chapter 6.3. Furthermore, the term "Resource" is used here in the sense of class rdfs:Resource. This is the superclass of all entities in an RDF based model. This can therefore refer to all kinds of entities, such as services, network resources, nodes, products, etc.

For example:

```
ex:T1
  a icm:Target ;
  icm:chooseFrom ( [ set:resourcesOfType ( ex:UserService ) ] )
.
ex:E1
  a icm:DeliveryExpectation ;
  icm:target ex:T1 ;
  icm:deliveryType ex:VideoService
.
```

This example specifies that a video service shall be delivered. More specifically it defines that the resource used for the expected delivery shall be an instance of class `ex:VideoService`. The target allows the intent handler to choose resources from the set of all user services, which are instances of class `ex:UserService`.

6.2. Description of resource to be delivered: icm:deliveryDescription

It is possible to refer to external artifacts that contain a detailed description of the expected delivery. The property `icm:deliveryDescription` refers to the resource that contains this description. This can for example be a service description document allocated in a service catalog.

For example:

```
ex:T1
  a icm:Target ;
  icm:chooseFrom ( [ set:resourcesOfType ( ex:UserService ) ] )
.
ex:E1
  a icm:DeliveryExpectation ;
  icm:target ex:T1 ;
  icm:deliveryType ex:VideoService ;
  icm:deliveryDescription ex:VideoServiceSpecGold ;
.
```

This example uses the property `icm:deliveryDescription` to specify that the resource `ex:VideoServiceSpecGold` contains more details. This example also specifies that the target resource used for delivering the specified service shall be of type ex:VideoService. This demonstrates that both properties can be used in combination. Consequently, the expectation evaluates to compliant if the chosen target resource is of type `ex:VideoService` and it complies to the servcie description `ex:VideoServiceSpecGold` as well.

The use case shown in this example typically involves a provisioning of the described service to be executed. The task of the intent handler would be to find a service resource to be used and therefore be considered as target. If none is available, the intent handler can initiate the provisioning and use the specified service description artifact as input. The expectation would indicate compliance as soon as the provisioning is successful, a new service instance is created and selected as target.

6.3. Target selection: icm:chooseFrom

An intent handler can select resources most suitable to use for delivering on the requirements. This allows an intent handler to optimize operation. However, a pre-determined target through ennumerating resource instances in the target defintion does not allow any freedom to make a selection. It prescribes the exact resources the requirements within the intent shall be applicable for.

The property `icm:chooseFrom` provides a container of resources in its range. These are the resources from which a choice can be made. The intent handler can choose a single resource or several depending on the use case specific needs. These reeouces would then be considered additions to the target assigned to the delivery expectation.

The choice of target resources is executed by the intent handler. The containers in the function arguments represent the set of resources to choose from. Any subset can be chosen. The intent manager will make a choice according to the type and number of resources suitable for meeting the requirements associated with the target through expectations the target is assigned to.

For example:

```
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice1 ;
.
ex:E1
  a icm:DeliveryExpectation ;
  icm:target ex:T1 ;
  icm:deliveryType ex:Slice ;
  icm:chooseFrom [ rdfs:member ex:Slice1, ex:Slice2, ex:Slice3, ex:Slice4 ]
.
```

This example defines the target `ex:T1` and uses it as target for the delivery expectation `ex:E1`. The delivery expectation states that something of class `ex:Slice` shall be delivered and that it shall be chosen from the set of four slice instances. As the target prescribes to use `ex:Slice1`, the intent handler has no choice than to use this instance. But additionally it can choose using any of the other three slice instances mentioned by the `icm:chooseFrom` property as well. Any additional choices would also be added implicitly to the target `ex:T1`. An respective target report within an intent report would state the complete set of resources used as target including the choices.

The set of resources to choose from can be determined by an abstract function that describes the criteria and conditions for members. For example:

```
ex:T1
  a icm:Target ;
.
ex:E1
  a icm:DeliveryExpectation ;
  icm:target ex:T1 ;
  icm:chooseFrom [ set:resourcesOfType ( ex:VideoService ) ]
.
```

This example demonstrates the choice of target members from any knwon video service. The target is empty by default. This means its members are solely defined by the choice an intent handler is making based on the `icm:chooseFrom` property. This property uses an conainer with memebers that are defined by the `set:resourcesOfType` function.

© TM Forum 2024. All Rights Reserved.

When using abstract functions or allowing the intent handler to choose target members, it is possible that no resources are found that match the criteria and the target container stays empty. An expectation with an empty or missing target defaults to not compliant. This might happen for example, if the intent handler has initiated the provisioning of a new resource to be used and therefore selected as target. Until the new resource is deployed, the target would be enpty becasue a choice cound not be made yet.

7. Property Expectation

The expectation class `icm:PropertyExpectation` allows defining conditions and goals based on observed system state and measurements. A typical use of this expectation would be requireing that a metric observed for the target shall show a specified value. For example, the availability of a targeted network node shall be above 99.99%, its average power consumption shall be below 1 kW and its operational state shall be "normal" or "standby". Only if the targeted resource meets these requirements it shall be considered compliant.

A property expectation also influences the selection of target resources. The intent handler would not only choose a resource that meets functional requirements, but it should also be expected to meet state and performance goals.

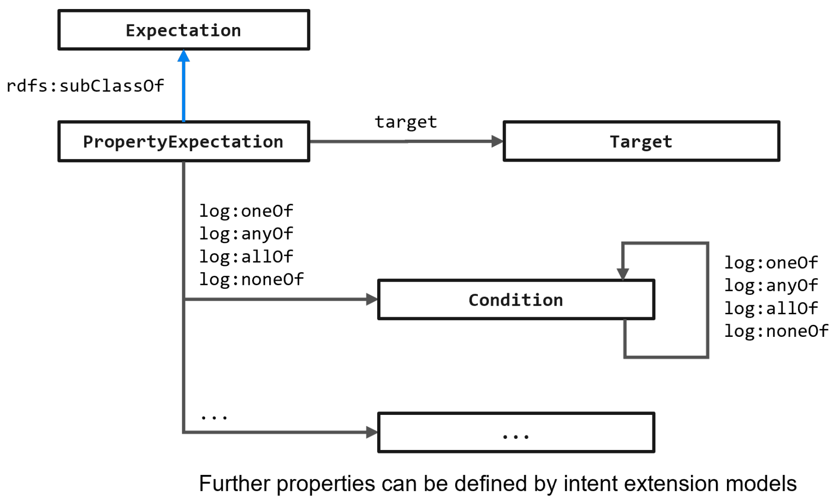

Figure 4: Property Expectation

The requirements of a property expectation are typically defined using conditions. This allows us to specify the exact logical evaluation tree to be used for determining compliance from observed values.

Property expectations depend on the availablilty of observable properties and a measurement infrastructure that provides value updates. Examples are metrics, key performance indicators (KPI) or resource states. They are domain and use case specific and their defintion is therefore not in scope of the generic vocabulary defined by the intent common model. Domain specific intent extension models would specify them. This means, a federation of the intent common model with domain specific models that contain a list and defintion of available observable metrics would provide an ontology of domain specific intent. This would be the base of domain specific intent handlers and the expression of domain specific requirements through property expectations.

For example:

```
ex:T1
  a icm:Target ;
  rdfs:member ex:RBS1
.
ex:E1
```

```
  a icm:PropertyExpectation ;
  icm:target ex:T1 ;
  log:allOf ( ex:C1 ex:C2 )
.
ex:C1
  a log:Condition ;
  quan:atLeast ( ex:Availability
                "0.9999"^^xsd:decimal
              )
  quan:atMost  ( ex:PowerConsumption
                "1.0kW"^^xsd:Quantity
              )
.
ex:C2
  a log:Condition ;
  set:intersects ( ex:RBSstate
                  [ a rdfs:Container ;
                    rdfs:member ex:RBSstate_normal ;
                    rdfs:member ex:RBSstate_standby ]
                )
.
```

This example expresses requirements for the resource `ex:RBS1`, which represents an individual radio base station site. This targeted resource has to meet the conditions associated with it through the property expectation `ex:E1`. Two conditions are defined. The first condition `ex:C1` specifies availability and power consumption goals. The second condition ex:C2 specifies that the state of the RBS system shall be either "normal" or "standby". In this example the states are represented by the individuals `ex:RBSstate_normal` and `ex:RBSstate_standby`. Furthermore, the observed state of the targeted RBS system is represented as member of the container `ex:RBSstate`. The condition is expressed using the set logic function `set:intersects`. It is "true" if and when the containers in its arguments share members. The first argument is the container of observed states and the second is the container of wanted states. This effectively expresses that the condition is met if the onserved state is either "normal" or "standby". If all three conditions are met, the target is considered compliant to the expectation.

8. Reporting Expectation

The expectation class `icm:ReportingExpectation` is used to specify under which conditions the intent handler is required to create and distribute an intent report. Intent reporting is primarily a push mechanism steered by reporting expectations. If and when the intent owner requires a different reporting scheme, it would modify the intent with a different reporting expectation.

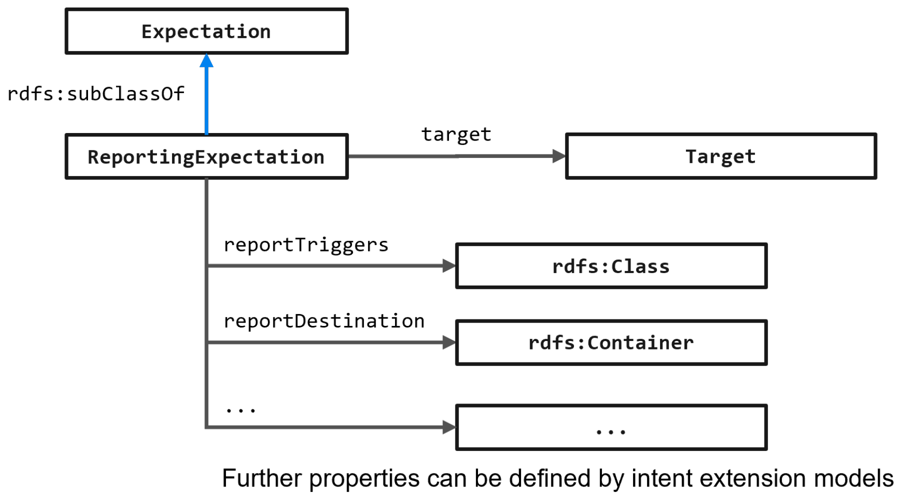

Figure 5: Reporting Expectation

An intent handler is compliant to a reporting expectation as long as all required reports could be generated and successully distributed. This means an intent can become degraded if reporting fails. As soon as reports are sent again successfully and the backlog of reporting is cleared, it would be complaint again. This means that reporting is yet another requirement that the intent owner is asking the intent handler to fulfill. As reporting closes the control loop it is essential for overall coordinated operation of the intent managed system. A loss of reporting is therefore a serious issue and its status of a requirement recognizes this.

Using multiple reporting expectations would allow us to distinguish multiple reports with individual scope, content and triggering event. It is, for example, possible to ask for a report about all elements of the intent when the system is getting degraded. If the system complies again a shorter report might be sufficient.

This document only discusses the expression of the reporting expectation, becasue it is part of the intent expression. The structure and content of intent report objects is in scope of the related document TR290B.

8.1. Reporting Scope

The scope of reporting is determined by the target of the reporting expectation. The target container can refer to any element of the intent. This refers to instances of the classes `icm:Intent`, `icm:Expectation`, `icm:Target`, `log:Condition`, `icm:Context` and `icm:Information` as well as their subclasses. These classes are defined as subclasses of icm:IntentElement.

© TM Forum 2024. All Rights Reserved.

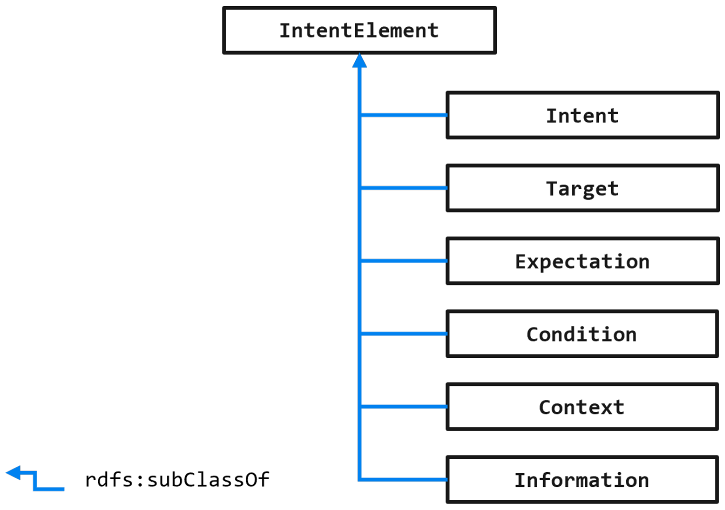

Figure 6: Intent Elements

Intent extension models might introduce further elements of intent to be included in an intent report. What an intent report about each of these elements would contain is specified in TR290B.

For example:

```
ex:Intent1
  a icm:Intent
.
ex:T1
  a icm:Target
.
ex:T2
  a icm:Target ;
  rdfs:member ex:Intent1, ex:T1, ex:E1, ex:E3, ex:C1
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1
.
ex:E2
  a icm:PropertyExpectation ;
  icm:target ex:T1
.
ex:C1
  a log:Condition
.
ex:C2
  a log:Condition
.
ex:E3
  a icm:ReportingExpectation ;
  icm:target ex:T2
.
```

This example shows an intent with multiple expectations and conditions. The expectation ex:E3 is a reporting expectation and the required reporting scope is determined by the target ex:T2. This target has five members referring to the intent itself, one of the property expectations, its target, one of the conditions and the reporting expectation. This means that the resulting intent report would need to contain explicit reporting about ex:Intent1, ex:T1, ex:E1, ex:E3 and ex:C1, but not about ex:T2, ex:E2 and ex:C2.

8.1.1. Reporting scope definition with icm:intentElements

The function `icm:intentElements` represents a container of all individual intent element resources used to express requirements for the intent elements given as arguments. This includes the given intent element object itself and its parents. This function can be used to define the reporting scope for an entire subtree of the intent. It represents all intent elements in the subtree that contains the function argument.

If used, for example, with an intent object as argument, the function `icm:intentElements` represents all individual intent elements used within this intent. If the argument is for example an individual expectation, the function would represent all intent elements used to express the requirements of this expectation, but not other expectations. For example:

```
ex:Intent1
  a icm:Intent ;
  log:allOf ( ex:E1 ex:E2 ex:E3 )
.
ex:T1
  a icm:Target
.
ex:T2
  a icm:Target ;
  icm:intentElements ( ex:E1 )
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1 ;
  log:allOf ( ex:C1 )
.
ex:E2
  a icm:PropertyExpectation ;
  icm:target ex:T1
  log:allOf ( ex:C1 ex:C2 )
.
ex:C1
  a log:Condition
.
ex:C2
  a log:Condition
.
ex:E3
  a icm:ReportingExpectation ;
  icm:target ex:T2
.
```

This example uses the the function icm:IntentElements to define the scope of reporting by specifying the members of the target of the reporting expectation `ex:E3`. The function argument is the expectation `ex:E1` indicating that the report shall be focussed on this expectation. As a result the target `ex:T2` has `ex:E1` as member. The intent object `ex:Intent1` is member, because it is the parent of `ex:E1`. The target `ex:T1` is member, becasue it is the target of `ex:E1`. The condition `ex:C1` is member, becasue it is used within `ex:E1`. Consequently, the resulting intent report will focus just on these intent elements.

8.1.2. Blank nodes

It is not possible to request an explicit individual report for an element that is expressed by a blank node in the Intent. This means that the expression of an intent element as blank node in the intent should be avoided, if the requester of an intent report requires explicit deteiled reporting about it. However, the summarized effects on the overall intent status the blank nodes contributed to would be visible in the reports fo its parent elements. For example, the resulting effects of a condition on an expectation regarding compliance are included in the compliance score of the expectation.

8.2. Report Content

The content of the intent report refers to the information provided about the intent itself and its elements in scope. What information is expected to be added to an intent report depends on the classes and subclasses of the reporting expectaion used to request is generation. This means there is a defined set of report content associated with the base class `icm:ReportingExpectation` and additional information would be added if the report is required using sub-classes of `icm:ReportingExpectation`.

The intent common model specifies the base class icm:ReportingExpectation and its sub-class `icm:ObservationReportingExpectation`. Further sub-classes can be added to the itnent ontology by intent extension models. The detaileed description of the content and information wihin an intent report is provided in part B of the intent common model specification [TR290B].

It is possible to add distinct reporting expectation individuals to an intent for various subclasses of reporting expectations. This is useful if different content is wanted, for example, for distinct reporting scopes or for different reporting triggers. However, if two intent reports are due for the same destination at the same time, an intent handler typically combines all details to be reported into a single report. Therefore, if only additional content is wanted, a single reporting expectation instance can be declared to be in instance of several reporting expectation classes. For example:

```
ex:E2
  a icm:ObservationReportingExpectation ;
  a iv:ValidityReportingExpectation ;
  a ut:UtilityReportingExpectation ;
  icm:target ex:T1 ;
  icm:reportDestination [ a rdfs:Container ;
                        rdfs:member ex:IMF1 ]
  icm:reportTriggers [ a rdfs:Container ;
                     rdfs:member imo:IntentRejected ;
                     rdfs:member imo:IntentAccepted ;
                     rdfs:member imo:Degrades ;
```

```
                    rdfs:member imo:Complies ]
.
```

This example defines that the intent report shall be sent to `ex:IMF1` and it shall be generated if any of the four specified intent handling events occurs. The reporting expectation is simultaneously an instance of the classes icm:ObservationReportingExpectation defined in the itnent common model, iv:ValidityReportingExpectation defined in the intent validity extension model and ut:UtilityReportingExpectation defined in the utility intent extension model. This means, the intent report to be generated shall include reporting about observations, validity and utility in addition to the basic intent report content.

8.2.1. Basic intent report content

Basic intent reporting content refers to the information provided within an intent report that is generated from icm:ReportingExpectation. As all specializations of intent reports are sub-classes of icm:ReportingExpectation, this is a common set of information to be added to all intent reports. This includes:

- A time stamp of intent report generation,
- A sequence number of the individual report,
- A reference to the intent element reported about,
- The intent handling and intent update state at the time of report generation,
- Report objects for all intent elements in scope,
- Compliance information about the intent and each intent element in scope when applicable,
- Reason objects that explain the compliance evaluation result

8.2.2. Reporting observation details

A basic intent report does not contain detailed observed or measured values for metrics used in the intent. This information can be requested with an observation reporting expectation. It is an instance of class `icm:ObservationReportingExpectation` and sub-class of `icm:ReportingExpectation`. For reports requested on behalf of observation reporting expectation, observation objects are added to the reports of intent elements that use observable metrics for expressing intent. The reporting section of the itnet common model specification [TR290B] describes the details of how this information would be represented in an intent report.

8.3. Report Destinations

The property `icm:reportDestination` is used with a reporting expecation and it specifies, where to send the required report. The object of the property is a resource that represents a wanted receiver of the intent report. When a report is generated, it would be sent to this receiver. For example:

```
ex:T1
  a icm:Target ;
  rdfs:member ex:Intent1 ;
.
```

```
ex:E1
  a icm:ReportingExpectation ;
  icm:target ex:T1 ;
  icm:reportDestination ex:IMF1 ;
  icm:reportDestination ex:Monitoring ;
  icm:reportDestination ex:Logging ;
.
```

This example defines a reporting expectation with a specification of report destinations. The resources `ex:IMF1`, `ex:Monitoring` and `ex:Logging` refer to entities, which are supposed to receive the intent report defined by this reporting expectation. The name of the resources in this example implies that another intent managment function instance, for example the owner of the intent as well as a monitoring system and a logging system shall receive the intent report.

The property `icm:reportDestinations` is used with a reporting expectation and it specifies where to sent the required report. The object of the property is a container with member resources that represent the wanted receivers of the intent report. When a report is generated, it would be sent to all receivers specified in the container. For example:

```
ex:E1
  a icm:ReportingExpectation ;
  icm:target ex:T1 ;
  icm:reportDestinations [ a rdfs:Container ;
                           rdfs:member ex:IMF1 ;
                           rdfs:member ex:Monitoring ;
                           rdfs:member ex:Logging ]
.
```

This example defines a reporting expectation with a specification of report destinations represented by members of a container. The resources `ex:IMF1`, `ex:Monitoring` and `ex:Logging` refer to functions, which are supposed to receive the intent report defined by this reporting expectation. This example is equivalent to the example given for the property ex:reportDestination. The only difference is that it is using containers of destinations rather than single statements per destination.

Using containers allows pre-defining sets of destinations and re-using these sets in multiple reporting expectations. Furthermore, the specification of report destinations can use any combination of multiple `icm:reportDestination` and `icm:reportDestinations` statements. The report would be sent to all of the destinations specified across these statements. For example:

```
ex:StandardDestinations
  a rdfs:Container ;
  rdfs:member ex:Monitoring ;
  rdfs:member ex:Logging
.
ex:E1
  a icm:ReportingExpectation ;
  icm:target ex:T1 ;
  icm:reportDestinations ex:StandardDestinations ;
```

© TM Forum 2024. All Rights Reserved.

```
  icm:reportDestination  ex:Dest1
.
ex:E2
  a icm:ReportingExpectation ;
  icm:target ex:T1 ;
  icm:reportDestinations ex:StandardDestinations ;
  icm:reportDestination  ex:Dest2
.
```

This example defines two reporting expectations `ex:E1` and `ex:E2` with a specification of respective report destinations. Two destinations are specified by the container `ex:StandardDestinations`. It is used as destination in both reporting expectations and assigned using the property `icm:reportDestinations`. Each of the reporting destinations also has a unique additional destination assigned using the `icm:reportDestination` property. The reports generated according to `ex:E1` need to be sent to the standard destinations specified by `ex:StandardDestinations` and to `ex:Dest1`. The reports generated according to `ex:E2` need to be sent to the standard destinations as well, and in addition it needs to be sent to `ex:Dest2`.

8.4. Event based reporting

Reports are generated and sent based on events of the intent handler. Intent handling events are defined in TR292B. It also specifies an intent handling and intent update state machine explaining the meaning of significant events in the intent handling process.

A reporting expectation defines which events shall initiate this intent report using the property `icm:reportTriggers`. Its range contains a container of events that shall trigger a report. For example:

```
ex:E2
  a icm:ReportingExpectation ;
  icm:target ex:T1 ;
  icm:reportDestination [ a rdfs:Container ;
                          rdfs:member ex:IMF1 ]
  icm:reportTriggers [ a rdfs:Container ;
                       rdfs:member imo:IntentRejected ;
                       rdfs:member imo:IntentAccepted ;
                       rdfs:member imo:Degrades ;
                       rdfs:member imo:Complies ]
.
```

This example defines that the intent report shall be sent to `ex:IMF1` and it shall be generated if any of the four specified intent handling events occurs.

8.5. Time based reporting

An intent owner might want to be regularly informed about the status of the intent. This is achieved by defining a new type of event with a time based triggering condition. The newly introduced event type is added to the report triggers.

For example: Generation of a new intent report if none was sent for 10 minutes:

```
ex:ReportEvent1
  a rdfs:Class ;
  rdfs:subClassOf imo:Event ;
  log:match ( [ t:after [ imo:timeOfLastReport ( ex:Intent1 ) ] ;
                t:hasDuration "PT10M"^^xsd:duration
              ]
              t:before
              [ t:hasBeginning imo:Now ]
            ) ;
  imo:eventFor ex:Intent1
.
ex:E2
  a icm:ReportingExpectation ;
  icm:target ex:T1 ;
  icm:reportDestinations [ a rdfs:Container ;
                           rdfs:member ex:IMF1 ] ;
  icm:reportTriggers [ a rdfs:Container ;
                       rdfs:member imo:IntentRejected ;
                       rdfs:member imo:IntentAccepted ;
                       rdfs:member imo:Degrades ;
                       rdfs:member imo:Complies ;
                       rdfs:member ex:ReportEvent1 ]
.
```

This example does use the `imo:timeOfLastReport` function, which provides the timestamp of when a report was last sent to for the intent. Consequently, every new intent report resets the time for issuing a new instance of `ex:ReportEvent1`. This means, a new instance of event ex:ReportEvent1 is only issued if more than 10 minutes have passed without an intent report being generated. The time based intent report trigger is only used if the time between two intent reports is getting longer than 10 minutes.

9. Context

The intent common model introduces the class `icm:Context`. Individuals of this class and its subclasses do not define additional requirements, but they specify contextual information about the requirements. Context therefore explains the requirement and its intended interpretation. It might introduce scope or constrain the requirement through side conditions. Context might therefore lead to a different compliance evaluation of the requirement or it can lead to different actions and solution strategies chosen by the intent handler.

For example, the requirement might be to deliver an available bandwidth of 10 Mbit/s to users. A context can additionally specify that this requirement is only needed on weekdays, for users belonging to a specific user group or within a certain geographical region. This would be validity, user and geographical context.

Another example would be a latency requirement of 10 ms. This shall be the requirement as long as the used bandwidth is below 10Mbit/s and above that bandwith only 15ms latency would be required. Context could be used to establish the bandwidth side condition. It constitutes a traffic context.

The intent common model defines the property `icm:context` as a generic way to assign any context object to a resource. Typically it is used to assign context to elements of an intent or intent report including, but not limited to intent, expectation, target and condition, as well as their reporting counterparts such as intent report and expectation report.

Different concerns and use cases for contexts are modeled through subclasses of `icm:Context`. Associated with these classes of context is vocabulary to express their specific concern. Most important is however that each class of context clearly defines how it affects the instance it is assigned to. For example, how it would impact the compliance evaluation through expectations and conditions. The following list shows a few typical examples of effects a context can have:

Validity: Depending on the conditions the context specifies, the individual it is assigned to would be ignored as if it and the requirement it represents would not exist. For example, an expectation would only participate in compliance evaluation if and as long as a condition set by the context is "true". This can be used to express validity of a requirement including conditional validity and temporal validity.

Scope: Depending on the context the compliance evaluation is limited to a defined scope of resources. An example would be a requirement that a slice delivers a user experience latency of 30ms. Context can express that this requirement needs to be applicable only for users of a defined user group and therefore not for all users served through the slice. This means, that this example works like an observation filter or selective measurement aggregator. Only a subset of all observations that were made according to the context are used in the compliance evaluation of the respective parameter.

Priority: Context can express a priority of a requirement compared to other requirements. It would encourage the intent handler to find solutions that fulfill this requirement while others can more easily be compromised.

Context is optional. If no context is given, the default interpretation of requirements would apply. If this is not wanted and the requirement is only supposed to be considered in defined situations and under specific conditions, or if the outcome of the requirement evaluation shall be different, context can be used. It would define the scope of situations the context is applicable and specifies how the intent handler shall modify its interpretation of the requirements.

© TM Forum 2024. All Rights Reserved.

10.Information

Within an intent, all requirements are specified with expectation objects and their properties in relation to a target. They are potentially further detailed using context. However, the intent owner might want to provide additional information that is not directly interpreted as requirements. Also the intent handler might want to provide additional information within an intent report. Doing so is the purpose of information objects.

The instances of class `icm:Information` provide additional information about resources and in particular intent and intent report elements. Information objects provide additional insights and context but without impacting the interpretation of requirements. The intent common model does not define specific types of information, but intent extension models can add subclasses of `icm:Information` to do so. The property `icm:information` allows to assign information objects to resources and in particular to elements of an intent and intent report. The intent common model does not define specific types of information properties, but intent extension models can add sub properties of `icm:information` with a range specific to respective subclasses of information. The information class and property constitute anchor points for additional definitions in intent extension models.

Information can address many concerns and there is no inherent restriction regarding its use. Information can convey operational and environmental details. It might help the intent handler to make the right choices and set priorities correctly. It is not interpreted as requirement or constraint. Therefore, information is not used to determine if resources are compliant to the intent requirements or not. The managed system or infrastructure cannot violate the specifications provided through information objects or become degraded because of it. Information can however influence the decisions and help prioritization between options. Therefore, it can have an effect on the actions taken and operational strategies chosen. In this respect, information might help with optimization.

Examples of potential use cases for information are:

Customer information: Provide details about which customer the intent originates from or which customer's requirements it represents.

Market information: In which market or market segment is the intent used.

Geographical information: In which country or region is the intent used.

Related Intent: Other individual intent with other intent handlers related to this intent, for example, by contributing to an end-to-end use case setup.

© TM Forum 2024. All Rights Reserved.

11.Life cycle aspects of Intent

11.1. Creation of Intent

Intent is arriving in the intent manager through the Intent Managment API TMF921. It carries an expression of additions to the ontology graph. Supported formats of intent expreessions are TURTLE, RDF/XML and JSON-LD. They are equivalent and describe additional sub-graphs and how they anchor within the exisiting ontology. The sub-graphs within the intent are instances of intent elements such as expectations, context, targets and conditions. It can also contain additional defintions such as new container specifications and new event classes.

11.2. Update of intent

Intent is updated by sending a new version of the intent over the intent API. If the update is accepted, the sub-graphs belonging to the original intent specification are altered and its content is replaced by the new sub-graphs from the update. If additional graphs were added in the update they are added to the ontology. If the update did not provide information about sub-graphs present in the original version of the intent, these sub-graphs are removed. Please note that the update contains a complete intent and any difference to the original intent needs to be established in the knowledge graph.

11.3. Removal of intent

If intent is removed over the intent API, the intent handler may remove all its components from the knowledge graph and stop considering the intent in its operation. This refers to all sub-graphs coming in the original intent, including all updates. It explicitly includes additional supplementatry defintions next to the actual intent elements. If the intent sub-graphs are actually removed, or otherwise taken out of operation and archived, this implmentation detail is out of scope of this specification.

© TM Forum 2024. All Rights Reserved.

12. Administrative Appendix

12.1. Document History

12.1.1. Version History

Version Number	Date Modified	Modified by:	Description of changes
3.0.0	07-Feb-2023	Alan Pope	Final edits prior to publication
3.1.0	11-Apr-2023	Alan Pope	Final edits prior to publication
3.2.0	15-Aug-2023	Alan Pope	Final edits prior to publication
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

12.1.2. Release History

Release Status	Date Modified	Modified by:	Description of changes
Pre-production	07-Feb-2023	Alan Pope	Updated to version 3.0.0
Pre-production	17-Mar-2023	Adrienne Walcott	Updated to Member Evaluated status
Pre-production	11-Apr-2023	Alan Pope	Updated to version 3.1.0
Pre-production	15-May-2023	Adrienne Walcott	Updated to Member Evaluated status
Pre-production	15-Aug-2023	Alan Pope	Updated to version 3.2.0
Pre-production	18-Sep-2023	Adrienne Walcott	Updated to Member Evaluated status
Pre-production	29-Feb-2024	Alan Pope	Updated to version 3.4.0
Production	26-Apr-2024	Adrienne Walcott	Updated to reflect TM Forum Approved status
Pre-production	03-May-2024	Alan Pope	Updated to version 3.5.0
Production	28-Jun-2024	Adrienne Walcott	Updated to reflect TM Forum Approved status
Pre-production	04-Jul-2024	Alan Pope	Updated to version 3.6.0
Production	30-Aug-2024	Adrienne Walcott	Updated to reflect TM Forum Approved status

© TM Forum 2024. All Rights Reserved.

12.2. 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*

© TM Forum 2024. All Rights Reserved.