TM Forum Technical Report

Intent Validity - Intent Extension Model

TR291A

Maturity Level: **General Availability (GA)**	Team Approved Date: 04-Jul-2024
Release Status: Pre-production	Approval Status: Team Approved
Version 3.6.0	IPR Mode: RAND

Notice

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

The intent validity model is an intent extension model. It contributes optional vocabulary to the TM Forum intent ontology used to specify requirement expressions that are only valid and considered to be required under specific conditions.

Introduction

By default, the requirements, goals, and constraints expressed within an intent have universal validity. This means an intent handler is expected to try to make the system comply to the intent unconditionally as long as the intent is not removed by the intent owner. But a requirement might not be needed to be fulfilled in specific situations, under certain conditions or for distinct periods in time. In default intent management based on the intent common models alone, the intent owner would be required to monitor when the intent, or individual expectations are needed and explicitly set and remove them. This can become a repetitive task involving many intent modifications and causing considerable communication traffic between intent handler and intent owner. Furthermore, this would not allow an intent handler to know about requirements becoming relevant ahead of time and prepare early to meet them immediately when needed.

With the intent validity model, the intent owner can delegate the validity control to the handler by specifying the detailed conditions of validity within the intent. The model allows defining complex validity schemes involving multiple types of conditions and alternative requirements with mutually exclusive validity. The intent owner only needs to update the intent if the requirement details themselves or the validity conditions require a change.

Knowing the validity conditions allows the intent handler to plan and prepare the system it operates ahead of time for meeting certain requirements. For example, it can start provisioning and reserving resources needed to meet a requirement at a reasonable time before the requirement becomes valid. This way, it is possible to avoid temporal degradation implied by known requirement changes, because the actions needed to reach compliance to the new requirements can be executed at the optimal point in time.

The intent validity model introduces the ontology with vocabulary and semantics for validity specifications within intent. It also defines how validity is represented in intent reports. Validity can be provided for an entire intent, or it can be associated with elements within an intent such as expectations and conditions. In this respect validity has an effect on the interpretation and evaluation of compliance.

Intent validity is introduced as context according to the general specification of context in the intent common model. Furthermore, the logical evaluation of validity is based on conditions specified in the Intent Common Model. Logical operators can be used to concatenate validity conditions. The conditions are interpreted as validity conditions when used within a validity context.

Scope

This document is part of the TR291 series specifying intent extension models published as optional models within the TM Forum Intent Ontology (TIO).

This document defines general purpose vocabulary extending the intent models for assigning validity to intents and intent elements.

Revision Information

This revision v3.6.0 of the intent validity model is an optional 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 intent validity 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).
Intent Validity Ontology	iv	http://tio.models.tmforum.org/tio/v3.6.0/IntentValidityOntology/	TM Forum	(This Model) Defines ontology model for expression of validity in various contexts.
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]

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

Table 1: Model references

The intent validity 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 validity model depends on select models from the TM Forum Intent Ontology such as the intent common model for definition of context and conditions. Further models, such as the Conditions and Logical Operators Ontology, the Quantity Ontology, the Set Operators model and the Function Definition Ontology can be used to express validity conditions. The Time Ontology in OWL is the base for expressing temporal conditions.

2. Validity definition and general interpretation

Validity is a property associated with an object. It determines if the object would be considered either valid or invalid. In the context of intent expression, the object's validity is assigned to can be, for example, the intent, expectations, conditions, targets, information and context. By default, and if no explicit information is specified or derived that states if an object is invalid or valid, it is considered to be valid by default.

If an object is invalid, the general idea is to consider it as if it were not present and neutralize its effect. This can mean different things depending on the nature of the object, the concerns it is addressing and its interpretation and impact within use cases. Consequently, the meaning and detailed interpretation and effect of validity needs to be defined specifically for distinct classes of objects. This document defines the use and interpretation of validity for elements and concerns defined in the Intent Common Model. This includes, for example, the effect of validity on compliance evaluation with intent, expectation and condition objects. Other intent extension models can use the vocabulary for validity expression as specified here, but they need to define the detailed interpretation and impact of validity within their concerns and contexts.

Validity is expressed by objects of class `iv:Validity`. Validity is a specialization of context defined in the intent common model. It is therefore modeled as subclass of class `icm:Context`.

Validity bears a truth value as a result of condition. The class `iv:Validity` is therefore also a specialization of conditions defined in the intent common model. It is a subclass of class `log:Condition`.

Validity has an extent in time referring to the time period the validity condition is true. The class `iv:Validity` is therefore defined as specialization of class `t:TemporalEntity` defined in the OWL time ontology.

3. Validity assignment

The validity associated with an object is determined by validity context assigned to this object. Validity is assigned using the property `iv:validIf`. As this property assigns a subclass of `icm:Context`, `iv:validIf` is defined as specialization and subproperty of the general property `icm:context`.

If multiple validity contexts are assigned to a single object, the overall validity of the object is determined by a logical conjunction of truth values from the individual validity context objects.

For example:

```
ex:V1
  a iv:Validity ;
  #...
.

ex:I1
  a icm:Intent ;
  log:allOf ( ex:E1 ) ;
  iv:validIf ex:V1
.

ex:E1
  a icm:PropertyExpectation ;
  log:allOf ( ex:C1 ) ;
  iv:validIf ex:V1;
  iv:validIf [ a iv:Validity
               #...
             ]
.
```

This example shows assignment of validity using explicitly named objects of type `iv:Validity` and the alternative of blank nodes of type `iv:Validity`. The first validity is defined as object `ex:V1`. This object is assigned as validity to the intent `ex:I1` using the property `iv:validIf`. A second validity is assigned to the expectation `ex:E1`. Here, a blank node is used rather than a named object. The example only shows the structure and use of validity assignments, but it leaves out the expression of detailed validity conditions.

3.1. Explicit validity inheritance

The property `iv:sameValidityAs` specifies that the validity of the subject shall be the same as the validity of the object. It therefore establishes an explicit inheritance of validity from other objects. The referenced object is not necessarily a parent object or an object in the same graph or namespace. It can, for example, be another intent or an expectation within another intent. If the object referenced does not have its own explicit validity definition, then its default or inherited validity is used.

If the referenced object does not exist or the validity cannot be determined for whatever reason, the inherited validity is considered to be "false", which means it is not valid. This default also applies if two objects create a logical loop, for example by referring to each other with iv:sameValidityAs.

For example:

```
ex:E1
  a icm:PropertyExpectation ;
  iv:sameValidityAs ex:E1;
  iv:validIf [ a iv:Validity
               #...
             ]
.
ex:E2
  a icm:PropertyExpectation ;
  iv:sameValidityAs ex:E1;
.
```

This example shows one expectation `ex:E1` with explicitly defined validity. The validity of another expectation ex:E2 follows the validity of ex:E1.

4. Validity expression

Validity is represented by a truth value with "true" meaning "valid" and "false" meaning "not valid". This means, validity can be defined based on any property and object that bears, determines or derives a truth value. Conditions and operators defined in the intent common model and other ontologies within the TIO allow the expression and assignment of implied of truth values. This constitutes a vocabulary basis of validity expression. However, any additional vocabulary introduced through separate models within or outside the TIO can also be used as long as it also directly or indirectly determines a truth value.

4.1. Validity conditions based on metrics

By using conditions as defined in the intent common models in combination with operators that provide a truth value, a validity context can express a broad variety of conditions to be interpreted as validity. It can express, for example that a measurable metric shall be below or above a certain threshold value. Other typical use cases are conditions based on sets. For example, the validity might depend on the classification of a user as being part of a specific user group, or a state shall be one of a select set of states.

The use of conditions for expressing validity is similar to the use of conditions for defining metrics based requirements in property expectations. But as the condition is contributing its resulting truth value to a validity context rather than an expectation, it expresses a validity side condition of the requirement rather than the requirement itself.

For example:

```
ex:V1
  a iv:Validity ;
  quan:smaller ( ex:NumberOfUsers
                 [ rdf:value 1000 ]
               )
.

ex:I1
  a icm:Intent ;
  iv:validIf ex:V1
.
```

This example shows how validity is specified based on the measurable metric `ex:NumberOfUsers`. The validity is assigned to the intent `ex:I1`. This means that the intent and therefore all requirements it specifies would be valid only if the number of users served is smaller than 1000 users. This establishing a side condition regarding when the requirements shall apply. Here the requirements are not required if the number of users is beyond a threshold. The managed system can therefore not fail to be compliant to this intent, as long as the number of users is 1000 or higher.

As the class iv:Validity is defined as subclass of log:Condition, validity objects can be directly used as conditions.

```
ex:V1
  a iv:Validity ;
  quan:smaller ( ex:NumberOfUsers
                 [ rdf:value 1000 ]
               )
.
ex:V2
  a iv:Validity ;
  log:noneOf ( ex:V1 )
.
ex:I1
  a icm:Intent ;
  iv:validIf ex:V1
.
ex:I2
  a icm:Intent ;
  iv:validIf ex:V2
.
```

This example shows how validity can be defined based on another validity. The validity of intent the first intent `ex:I1` is determined by `ex:V1`. The validity of the second intent is determined by ex:V2. Here, ex:V2 is defined based on ex:V1 using the logical operator log:noneOf. In this example, this practically expresses a negation. This means ex:V2 is valid if and only if ex:V1 is not. Here ex:V1 is directly used as condition object. This example therefore expresses that the requirements specified by intent ex:I1 shall apply if the number of users is smaller than 1000 and the requirements from intent ex:I2 shall apply otherwise. This example also demonstrates how validity allows specifying conditional alternative sets of requirements.

4.2. Temporal validity

Requirements might be needed within certain timeframes. This can be expressed through validity conditions based in date and time. The intent validity model proposes to use the OWL time ontology specified by W2C.

For example:

```
ex:V1
  a iv:Validity ;
  log:match ( [ imo:Now ]
              t:after
              [ t:hasBeginning [ a t:DateTimeDescription ;
                                 t:minute 0 ;
                                 t:hour 1 ;
                                 t:day 5 ;
                                 t:month 7 ;
                                 t:year 2023 ;
                                 t:timeZone
```

```
<https://www.timeanddate.com/time/zones/cet> ;
                        ]
          ]
         )
.
ex:I1
  a icm:Intent ;
  iv:validIf ex:V1
.
```

This example defines validity based on a temporal condition using the vocabulary of the time ontology in OWL referenced by the prefix "t:". The condition is "`true`", and the validity evaluates to "valid" if the current point in time is after a certain point in time. This means it defines validity with a start date and time. The current point in time is represented by `imo:now` defined in the intent management ontology. It is compared with the point in time defined using a `t:DateTimeDescription` and specifying 5th of July 2023 at the time 01:00 within the timezone. The condition is then based on a statement using `t:after` as predicate with the current time as subject and the wanted starting time as subject. If this statement is true the condition is true, and the validity context indicates "valid". It is assigned to an intent. This means the entire intent and all its requirements only need to be fulfilled within the stated time interval.

The example above only defines a start time of validity, but no end time. An intent associated with this validity would therefore be valid indefinitely once the start time is reached. A validity can also be defined with an end time using time intervals.

For example:

```
ex:V1
  a iv:Validity ;
  log:match ( [ imo:Now ]
              t:inside
              [ t:hasBeginning [ t:inXSDDateTimeStamp "2023-07-
05T01:00+01:00"^^xsd:dateTimeStamp ] ;
                t:hasXSDDuration "PT10M"^^xsd:duration ]
            )
.
ex:I1
  a icm:Intent ;
  iv:validIf ex:V1
.
```

This example defines validity based on a temporal condition using the vocabulary of the time ontology in OWL referenced by the prefix "t:". The condition is "`true`", and the validity evaluates to "valid" if the current time is inside a certain time interval. The current point in time is represented by `imo:now` defined in the intent management ontology. The time duration is specified using a temporal entity that starts at the 5th of July 2023 at the time 01:00 within the timezone UTC+1 and has a length in time of 10 minutes. The temporal entities used in this example are specified using the syntax of XML schema for expressing date and time stamps and time duration. The condition is then based on a statement using the predicate `t:inside`, with the current time as subject and the time interval as object. If and while this statement is true the condition is true and the validity context indicates "valid". It is assigned to an intent. This means

the entire intent and all its requirements only need to be fulfilled within the stated time interval.

It is possible to define extensive temporal validity schema and recurring validity time intervals.

For example:

```
ex:V1
  a iv:Validity ;
  log:match ( [ imo:Now ]
              t:inDateTime
              [ t:dayOfWeek t:Saturday, t:Sunday]
            )
.
ex:I1
  a icm:Intent ;
  iv:validIf ex:V1
.
```

This example defines validity based on days of the week. The intent is valid every Saturday and Sunday. Consequently, the requirements specified by this intent shall only apply on weekends.

It is also possible to combine temporal and other conditions for expressing validity.

5. Validity interpretation and impact on intent elements

This chapter specifies the detailed interpretation of validity for the objects of intent

5.1. Impact on compliance evaluation

Intent and Intent expectation are objects that bear a truth value interpreted as compliance to the requirements defined by them. If an intent or expectation is invalid it shall not contribute to the overall evaluation of compliance and in particular, it shall not lead to evaluation as not compliant or partially not compliant. This means compliance is assumed by default for invalid intent and expectation objects.

If an intent is invalid as a whole, none of its requirements are valid and therefore none of them need to be fulfilled. Consequently, the operated system would be evaluated as compliant to this intent mo natter of its state and achieved KPI as long the intent is invalid.

The validity of expectations is interpreted similarly to the intent, but only effecting the subset of requirements specified by the expectation. An invalid expectation therefore contributes the logical "true" into the conditions for compliance evaluation of the intent. This default interpretation of invalid expectations need to be considered when specifying compliance evaluation logic. For example, when using logical disjunction with multiple expectations, an invalid expectation would contribute "true" indicating compliance. It would contribute in the same way as it would be valid and all its requirements were fulfilled. This means an invalid expectation can cause the entire logical disjunction to evaluate to "true" and compliant.

For example:

```
ex:V1
  a iv:Validity ;
  a log:Condition ;
  quan:smaller ( ex:NumberOfUsers
                 [ rdf:value 1000 ]
               )
.

ex:I1
  a icm:Intent ;
  log:anyOf ( ex:E1 ex:E2 ) ;
.

ex:E1
  a icm:PropertyExpectation ;
  log:allOf ( ex:C1 )
.

ex:E2
  a icm:PropertyExpectation ;
  log:allOf ( ex:C2 )
  iv:validIf ex:V1
```

```
.
```

This example specifies intent `ex:I1` with compliance depending on the expectations `ex:E1` and `ex:E2`. The validity of `ex:E2` depends on the number of users served, and it is only valid if the number of users is smaller than 1000. In this example the compliance to the intent is determined by a disjunction of the two expectations. This means only one needs to show compliance to reaching overall compliance to the entire. This has the effect that overall compliance to the intent is always reached if the number of users is 1000 or greater. In this case the expectation `ex:E2` becomes invalid, and the system cannot be not compliant to this expectation anymore. But as the system is compliant to `ex:E2`, it becomes overall compliant to the intent irrespective of the still valid expectation `ex:E1`. It is important to consider these effects of validity on compliance logic when creating an intent. The effects of invalid requirements might not always be intuitive.

5.2. Validity of Conditions

Validity can be assigned to conditions. If a condition is invalid it defaults to the value "true" irrespective of how its condition details would evaluate.

This default for invalid conditions impacts the domain in which the condition is used. For example, if it is a condition used for compliance of expectations, an invalid condition would indicate compliance or contribute a partial compliance in into the compliance evaluation logic for this expectation. This is similar to the logical contributions of invalid expectations to the compliance evaluation of the intent as described in Chapter 5.1. However, conditions can also be used in other context and for other purposes than compliance. Nevertheless, the general rule applies that invalid conditions are "true" by default.

5.3. Validity of Targets and Containers

Targets determine what objects shall fulfill the requirements defined by an expectation. This means targets are essential for expectations to be meaningful. An expectation without a target or a target that is an empty container would define requirements that nothing and nobody needs to comply to.

The intent common model formally defines Targets are formally defined as containers of resources. It is a set of resources and target definition is therefore mainly concerned with operations to add or filter out members of the target container. Typical examples are

- the explicit enumeration of members,
- abstract description of conditions that a resource needs to fulfill to be considered to be a member of a target,
- Set operators such as union or intersection filter out members based on other containers.

A validity context can be assigned to a target or container. An invalid target or container is treated as empty set. Therefore, an invalid target or container will contribute an empty set in set operators and in targets of expectations with all effects an empty set would have. For example, an invalid target would leave an expectation without somebody or something that is required to fulfill it. If there is nothing that needs to meet the requirements, the logical consequence is that the respective expectation for this target is always fulfilled, and its compliance evaluation always determines compliance.

Validity used with targets and containers, can be used to conditionally change the resources that need to comply to the expectations. As such it allows defining alternative targets based on situation, operational context and time.

For example:

```
ex:V1
  a iv:Validity ;
  a log:Condition ;
  quan:smaller ( ex:NumberOfUsers
                 [ rdf:value 1000 ]
               )
.
ex:I1
  a icm:Intent ;
  log:anyOf ( ex:E1 ex:E2 ) ;
.
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice1
.
ex:T2
  a icm:Target ;
  set:resourcesOfType ( ex:Slice ) ;
  iv:validIf ex:V1
.
ex:T3
  a icm:Target ;
  set:union ( ex:T1 ex:T2 )
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T2 ;
  log:allOf ( ex:C1 )
.
ex:E2
  a icm:PropertyExpectation ;
  icm:target ex:T3 ;
  log:allOf ( ex:C1 )
.
```

This example defines multiple targets. The target ex:T1 contains just the slice individual `ex:Slice1`. The target `ex:T2` and its members are abstractly defined using the `set:resourcesOfType` operator. All resources of type `ex:Slice` are members,

which basically means all slices. This target is also subject to validity as defined by `ex:V1` based on the number of users served by the network. The example also defines target `ex:T3` as union of `ex:T1` and `ex:T2`. Furthermore, target `ex:T2` is used for the expectation `ex:E1` and target `ex:T3` is used for expectation `ex:E2`.

If the target `ex:T2` gets invalid, it defaults to an empty container. Consequently, the expectation `ex:E1` does not have a resource anymore that would need to fulfill its requirements. As target `ex:T3` is based on `ex:T2` in a union with `ex:T1`, an invalid `ex:T2` would mean that its elements are removed from `ex:T3`. This means that `ex:T3` would be left only with `ex:Slice1` contributed from `ex:T1` and this would be the only target for expectation `ex:E2` as long as `ex:T2` is invalid.

5.4. Validity of Context

Context can be also valid or invalid. This means a validity context can be applied to other contexts including other validity contexts. As any context has a particular effect on the object it is assigned to, the validity of the context conditionally activates or deactivates this effect. If a context is invalid, the intent handling falls back to the default that would be considered for the object if the context would not be provided or present in the intent expression.

For example, a context might introduce a specialization of a requirement. According to this example context the target would need to fulfill the requirement only for a certain group of users. If this context is invalid, the user group side-condition is removed and the default interpretation of the requirement applies. In this example this would mean that the requirement is applied irrespectively of user groups, and therefore it would be applicable for all users.

5.5. Validity of Information

Information objects provide additional hints and information that might be interesting for the intent handler. An information object that is invalid shall be ignored. The validity context declares that information is applicable or not. The use of validity context with information objects allows specifying alternative information and the conditions under which an alternative shall be considered or ignored.

6. Value assumed when invalid

The subject of a validity context might have an associated value. If the subject becomes invalid, the associated value shall not be used anymore. In some situations this can lead to errors in the interpretation of the model due to undefined values.

Using the property `iv:valueIfNotValid` allows setting a default value that shall be assumed for the associated value of the subject in case it becomes invalid. If an value to be assumed when invalid is not provided, the default associated value specified in Intent Management Elements [TR292A] shall be assumed. A default value and default combination method is defined for Boolean values, numeric datatypes including quantities and containers in [TR292A]. For other datatypes and classes or if the default value according to [TR292A] is not suitable, it can be specified explicitly using `iv:valueIfNotValid`.

For example:

```
ex:V1
  a iv:Validity ;
  a log:Condition ;
  quan:smaller ( ex:NumberOfUsers
                 "1000"^^xsd:Integer
               )
.
ex:C3
  a log:Condition ;
  log:allOf ( ex:C1 ex:C2 ) ;
  iv:validIf ex:V1 ;
  iv:valueIfNotValid "false"
.
```

This example defines a validity with the condition that the quantity `ex:NumberOfUsers` is smaller than "1000". This validity is assigned to the condition `ex:C3`. The associated boolean type value of the condition is determined by the `log:allOf` operator and its arguments. Furthermore, this example specifies, that the associated value shall be "`false`" if and while the condition `ex:C3` is invalid.

7. Reporting validity

This chapter specifies how validity would be represented in intent reporting. This includes the subscription to reports with reporting expectations as well as the intent report expression.

7.1. Validity Reporting Expectation

A validity reporting expectation is an instance of class `iv:ValidityReportingExpectation` within an intent. It is a subclass of `icm:ReportingExpectation`. A validity reporting expectation is chosen to request an intent report, if the intent report shall explicitly state the validity of intent elements.

For intent reports that are not generated from a validity reporting expectation, intent elements that were invalid at the time of reporting are excluded. Consequently, the intent report would not contain a respective report element for the invalid intent elements. This changes when the report is generated for a validity reporting expectation. Report elements are included for all intent elements in scope of reporting regardless of their validity. In addition, the validity status is explicitly stated.

If an intent report is generated for a validity reporting expectation it can contain `iv:validityStatus` properties associated with dedicated reports for intent elements. It provides the validity status at the time of report generation and regarding the intent element referred to with the property `icm:about`. The property `iv:validityStatus` assigns a boolean value. The value "true" means the intent element is valid and "false" means invalid.

The property `iv:validityStatus` can be left out for intent elements that are valid. If it is missing the intent report is interpreted accordingly. If the respective intent element is not valid, the property `iv:validityStatus` is needed to report this.

For example, an expectation report object within an intent report reports about an expectation object in the intent. The property `iv:validityStatus` of this expectation report shows "true", if the expectation reported about is valid.

7.2. Validity in reporting scope

According to the intent common model the scope of reporting refers to those elements of an intent expression that shall be addressed in an intent report. This includes, for example expectations and conditions. It can also include context and therefore also validity context.

The scope of an intent report is determined by the target of the respective reporting expectation. Also, individuals of class `iv:Validity` can be added to the reporting target container. This would result in receiving a respective validity report within the intent report.

For example:

```
ex:V1
  a iv:Validity ;
  a log:Condition ;
  quan:smaller ( ex:NumberOfUsers
                 [ rdf:value 1000 ]
```

```
            )
.
ex:I1
  a icm:Intent ;
  log:anyOf ( ex:E1 ex:E2 ) ;
.
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice1
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T2 ;
  log:allOf ( ex:C1 ) ;
  iv:validIf ex:v1
.
ex:C1
  a log:Condition ;
  quan:smaller ( ex:NumberOfUsers
                 [ rdf:value 1000 ]
               )
.
ex:T2
  a icm:Target ;
  rdfs:member ex:I1 ;
  rdfs:member ex:E1 ;
  rdfs:member ex:C1 ;
  rdfs:member ex:V1
.
ex:E2
  a icm:ReportingExpectation ;
  icm:target ex:T2 ;
.
```

This example defines a validity context `ex:V1`, which is used to determine the validity of expectation `ex:E1`. The reporting shall include a detailed report about the validity context. To initiate the generation of a report for the validity context `ex:V1`, it is added as member to the target container `ex:T2`, which is the target of the reporting expectation `ex:E2`. This means that reporting about `ex:V1` is in scope of reports generated according to `ex:E2`.

7.3. Validity report

A validity report is an instance of class `iv:ValidityReport`, which is a subclass of `icm:ContextReport` as defined in the intent common model. A validity report is exclusively used to provide reporting for instances of class `iv:Validity`.

The property `icm:about` as defined in the intent common model is also used to refer to the individual validity reported. This means instances of class `iv:ValidityReport` can be in the domain of icm:reportsAbout and instances of `iv:Validity` can be in its range.

The property `icm:result` as defined in the intent common model states the result of a logical evaluation at the time of report generation. If used with a validity report, it states the result of the validity condition.

The property `iv:isValid` is used to state the validity status of the object in its domain. This means it states the validity at the time of report generation. The range of `iv:isValid` contains a Boolean literal value. It is set to "`true`", if the object is valid and "`false`" if not. The property `ex:isValid` can be applied to validity report objects, but also to an intent report and report elements within an intent report. It can therefore state, for example, if an intent, an expectation, a condition, a target or a context were valid.

The property `iv:isValid` can be optionally used in the reports about any intent element. If `iv:isValid` is omitted from a report, the element that is reported about is by default considered to be valid.

The property `iv:isValid` shall be used in a report about an intent element, if a validity context is either directly assigned to the intent element or it is indirectly determined through validity inheritance. This means, it shall be used if the validity of the intent element reported about is not always "true". This can be the case if a validity context is explicitly assigned to an intent element using the `iv:validIf` property, or if validity is derived from a parent element that has its validity determined by a validity context.

For example:

```
ex:I1R1
  a icm:IntentReport ;
  icm:about ex:I1
  icm:result "true"
.
ex:V1R1
  a iv:ValidityReport ;
  icm:about ex:V1 ;
  icm:result "true"
.
ex:T1R1
  a icm:TargetReport ;
  rdfs:member ex:Slice1
.
ex:E1R1
  a icm:ExpectationReport;
  icm:targetReport ex:T1R1 ;
  icm:about ex:E1 ;
  icm:result "false" ;
  iv:isValid "true" ;
.
ex:C1R1
```

This example shows an intent report that includes reporting about validity of intent elements. It includes and validity report stating that the result of validity evaluation is "`true`". This means that all intent elements that receive their validity from the reported about validity context were valid. The example also reports about the expectation `ex:E1` with the expectation report `ex:E1R1`. In this example, the validity context `ex:V1` determines the validity of expectation `ex:E1`. Consequently, the respective expectation

reports states that the `ex:E1` is valid using the property `iv:isValid`. The intent `ex:I1` is also valid, but there is no explicit statement about this in the intent report `ex:I1R1`. This is not necessary, because there were no validity contexts associated with the intent and therefore the intent is always valid.

7.4. Interpretation of reported result and validity

It is important to note, that in intent reporting `iv:isValid` is used in combination with `icm:result`. Both are stating a truth value that ultimately originates in conditions and their combined evaluation with logical operators, but their meaning and interpretation and therefore their use in intent reporting is different.

The property `icm:result` states the Boolean value associated with an object. For example, intent, expectation, condition and validity context objects have a Boolean value associated with them, but the interpretation of this boolean value can be different and depends on the type of object. The Boolean value associated with an intent or expectation, is interpreted as compliance of the system to requirements. In contrast, the boolean value associated with a condition is neutral and does not bear a particular interpretation. The interpretation of the boolean value of a condition is determined by the type of subject of properties that assign a condition object to this subject. For example, a logical operator with an expectation in its subject and a condition in its object states that the condition evaluation would contribute to the Boolean value of the expectation, which is then interpreted as compliance, because it is an expectation type subject. If the same condition object is assigned by the same logical operator to a validity context, it is interpreted as contribution to validity logic, because that is the interpretation inherent to validity context. In all these cases the property icm:result would state the Boolean value associated with an object without implying a particular interpretation. It only provides the value associated with the intent element reported about. The interpretation depends on the type of object it is assigned to.

In contrast, the property iv:isValid has a boolean value that is always interpreted as validity of the subject. For example, `icm:result` states "`false`" for an expectation and `iv:isValid` might state "true" for the same expectation. This means the conditions, properties and logical operators that contribute a boolean value directly to the expectation were overall evaluated as "`false`". This is interpreted as not compliant because it is an expectation. Nevertheless, this expectation is valid as stated by `iv:isValid`. This is the result of conditions, properties and logical operators, that contribute a boolean value to a validity context. This validity context is assigned to the expectation using the property `iv:validIf`.

This means that `icm:result` states the inherent boolean value of a property. Additional Boolean values associated with an intent element and determined by contexts would be stated by context specific additional properties such as `iv:validIf`.

8. Validity Operators

8.1. Validity Check

The intent validity model defines an operator property that refers to the validity status of resources.

```
<validity State> = iv:validityOf (  <rdf:Resource> ... )
```

The intent validity model introduces the property `iv:validityOf` that delivers a Boolean value based on the validity state of objects. The range of `iv:validityOf` is a container of arguments, which are resources for which the validity shall be checked. If all arguments are valid, then the result of `iv:validityOf` is "`true`", if any resource used as argument is not valid the result of `iv:validityOf` is "`false`".

9. Validity events

Validity is an inherent state of a resource and in particular of intent and elements within an intent. The intent validity model introduces new subclasses of `imo:IntentHandlingEvent` that would be raised by the intent handler to indicate a change of validity for resources.

Events of type `iv:ValidityChange` indicate that the validity of a resource did change from valid to invalid or from invalid to valid. Events of class `iv:ValidityChangeValid` and `iv:ValidityChangeInvalid` indicate that a resource became valid or invalid respectively.

These events are triggered for validity changes of any resource. An instance of these classes represents an individual event. The property `imo:eventIssuedFor` states which particular resource had a change of validity state.

A validity change event is issued for newly created intent in the intent handler if the initial validity state is "invalid". If the initial validity state for a new intent or its elements is "valid", no validity change event is issued.

9.1. Custom scoped validity events

It is possible to introduce new custom event classes regarding validity change as subclasses of `iv:ValidityChange`, `iv:ValidityChangeValid` or `iv:ValidityChangeInvalid`. This follows the definition of custom events as described in the Intent Management Ontology TR292B. The property imo:eventFor defines the scope of the custom validity event by specifying the resources for which a validity change shall be monitored and reported by the event.

For example:

```
ex:ChangeInvalid_E1
  rdfs:subClassOf iv:ValidityChangeInvalid ;
  imo:eventFor ( ex:E1 ex:E2 )
.
```

This example assumes an intent with expectations `ex:E1` and `ex:E2`. It defines a new subclass of `iv:ValidityChangeInvalid`, which is a subclass of `imo:IntentHandlingEvent`. The two expectations `ex:E1` and `ex:E2` are in scope of this event class as specified by `imo:eventFor`. This means an individual of this event would be created if the validity status of the expectations `ex:E1` or `ex:E2` is becoming invalid.

10.Administrative Appendix

10.1. Document History

10.1.1. Version History

Version Number	Date Modified	Modified by:	Description of changes
1.0.0	31-Mar-2022	Alan Pope	Final edits prior to publication
1.1.0	01-Jun-2022	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

10.1.2. Release History

Release Status	Date Modified	Modified by:	Description of changes
Pre-production	31-Mar-2022	Alan Pope	Initial Release
Pre-production	02-May-2022	Adrienne Walcott	Updated to reflect TM Forum Member Evaluated Status
Pre-production	01-Jun-2022	Alan Pope	Updated to v1.1.0
Pre-production	04-Jul-2022	Adrienne Walcott	Updated to reflect TM Forum Member Evaluated status
Pre-production	15-Aug-2022	Alan Pope	Updated to v3.2.0
Pre-production	18-Sep-2023	Adrienne Walcott	Updated to Member Evaluated status
Pre-production	29-Feb-2024	Alan Pope	Updated to v3.4.0
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

10.2. Acknowledgments

Team Member (@mention)	Company	Role*
Jörg Niemöller	Ericsson	Author, Project Co-Chair
Kevin McDonnell	Huawei	Project Co-Chair

Team Member (@mention)	Company	Role*
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*

11.Appendix A: Vocabulary Reference

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

11.1. isValid

The property `iv:isValid` is used to state the validity status of its object. It is used in intent reports. This means it states the validity at the time of report generation. The range of `iv:isValid` contains a Boolean literal value. It is set to "`true`", if the object is valid and "`false`" if not.

Instance of: `rdf:Property`
Range: `xsd:Boolean`

11.2. sameValidityAs

The property `iv:sameValidityAs` specifies that the validity of the subject shall be the same as the validity of the object.

Instance of: `rdf:Property`

11.3. validIf

The property `iv:validIf` assigns validity context to objects.

Instance of: `rdf:Property`Sub property of: `icm:contextRange: iv:Validity`

11.4. Validity

The class `iv:Validity` is a subclass of `icm:Context`. It represents and determines the validity of resources it is associated with using the property `iv:validIf` .

Instance of: `rdfs:Class`
Subclass of: `icm:Context`Subclass of: `t:TemporalEntity`

11.5. ValidityChange

The class `iv:ValidityChange` is a subclass of `imo:intentHandlingEvent` and represents events issued if the validity status of a resource is changing.

Instance of: `rdfs:Class`
Subclass of: `imo:IntentHandlingEvent`

11.6. ValidityChangeInvalid

The class `iv:ValidityChange` is a subclass of `iv:ValidityChange` and represents events issued if the validity status of a resource is changing to "invalid".

Instance of: `rdfs:Class`
Subclass of: `iv:ValidityChange`

11.7. ValidityChangeValid

The class `iv:ValidityChange` is a subclass of `iv:ValidityChange` and represents events issued if the validity status of a resource is changing to "valid".

Instance of: `rdfs:Class`
Subclass of: `iv:ValidityChange`

11.8. validityOf (<rdf:Resource> ...)

The property `iv:validityOf` is a function that states the validity status of its arguments. If all arguments are valid, then the result of `iv:validityOf` is "true", if any resource used as argument is not valid the result of `iv:validityOf` is "`false`".

Instance of: `fun:Function`Result type: `xsd:Boolean`
Arity: None or any number of arguments
Argument types: any resource

11.9. ValidityReport

Instances of class `iv:ValidityReport` are used in intent reports to provide reporting about validity context.

Instance of: `rdfs:Class`
Subclass of: `icm:ContextReport`

11.10. ValidityReportingExpectation

A validity reporting expectation is an instance of class `iv:ValidityReportingExpectation` within an intent. A validity reporting expectation is chosen to request an intent report, if the intent report shall explicitly state the validity of intent elements.

Instance of: `rdfs:Class`
Subclass of: `icm:ReportingExpectation`

11.11. validityStatus

If an intent report is generated for a validity reporting expectation it can contain `iv:validityStatus` properties associated with dedicated reports for intent elements. It provides the validity status at the time of report generation and regarding the intent

element referred to with the property `icm:about`. The property `iv:validityStatus` assigns a boolean value. The value "true" means the intent element is valid and "false" means invalid.

Instance of: `rdf:Class`
Range: `xsd:boolean`

11.12. valueIfNotValid

The property iv`:valueIfNotValid` allows specifying a function result that shall be provided as a default for invalid functions.

Instance of: `rdf:Property`

11.13. Vocabulary

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

Instance of: `rdfs:Container`