TM Forum Technical Report

Intent Management Elements

TR292A

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

Notice

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

The intent management elements model is an ontology within the TM Forum intent ontology. It introduces basic terminology used to describe intent based operation.

Introduction

The TM Forum Autonomous Networks Project introduces concepts around the use of intent within an autonomous network. For example, it introduces an architecture and functions to participate in intent based operation, roles within intent life cycle management and the intent API. The intent management elements model specifies an ontology with vocabulary to describe these concepts as part of intent modelling.

Scope

This document describes parts of the intent management ontology. Its focus is the introduction of general terminology used to describe intent based operation and the functions participating in it.

Revision Information

This revision v3.6.0 of the intent management elements ontology model is part of the TM Forum Intent Ontology (TIO) v3.6.0.

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

- Minor editorial corrections.

1. Notation and Dependencies

The Intent management elements model depends on the following models:

Model	Prefix	Namespace	Published by	Purpose in the model
Intent Management Elements	imo	http://tio.models.tmforum.org/tio/v3.6.0/IntentManagementOntology	TM Forum	Defines basic vocabulary and concepts of intent based operation. This document specifies vocabulary about intent management function and their roles, as well as the types of intent models within the TM Forum Intent Ontology (TIO).
RDF version 1.1	rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	W3C	Providing fundamental modeling artifacts
RDF Schema 1.1	rdfs	http://www.w3.org/2000/01/rdf-schema#	W3C	Providing fundamental modeling artifacts
Intent Common Model	icm	http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/	TM Forum	The intent common model is used here for providing examples
Examples	ex	http://www..example.org/	IANA	Reserved domain name for examples

Table 1.1: Model references

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

2. Intent management function roles

The technical architecture of the autonomous network introduces the intent management function [ig1251] [ig1253]. Instances of the intent management function, aka. "Intent Managers", communicate through the Intent Management API [tmf921] and exchange intent and intent reports with each other. They form an intent based control loop.

An intent manager can be the source of intent. This means it needs to communicate requirements to another system, sub-system or operational domain, and it uses intent to express these requirements. It would then communicate the intent to other intent managers using the intent API. Furthermore, this intent manager would further manage the life cycle of the individual intent object it has created. This means it can update the intent as needed and delete it once the requirements are not needed anymore. An intent manager in this role in the life-cycle of an intent is referred to as "intent owner".

An intent manager can also receive intent over an intent API. This means it is asked to fulfil the requirements expressed within the intent within its domain of responsibility. Typically, this entails the coordination of actions with other management functions within its domain. An intent manager in this role is referred to as "intent handler". An intent handler would also create intent reports to inform the intent owner about success and progress in meeting the requirements.

Every individual intent has exactly one owner and one handler. This implies that an intent manager can assume the role of intent owner for some intents and the role of intent handler for others.

The intent management elements model introduces vocabulary to identify intent management functions and their roles within the life cycle management of intent.

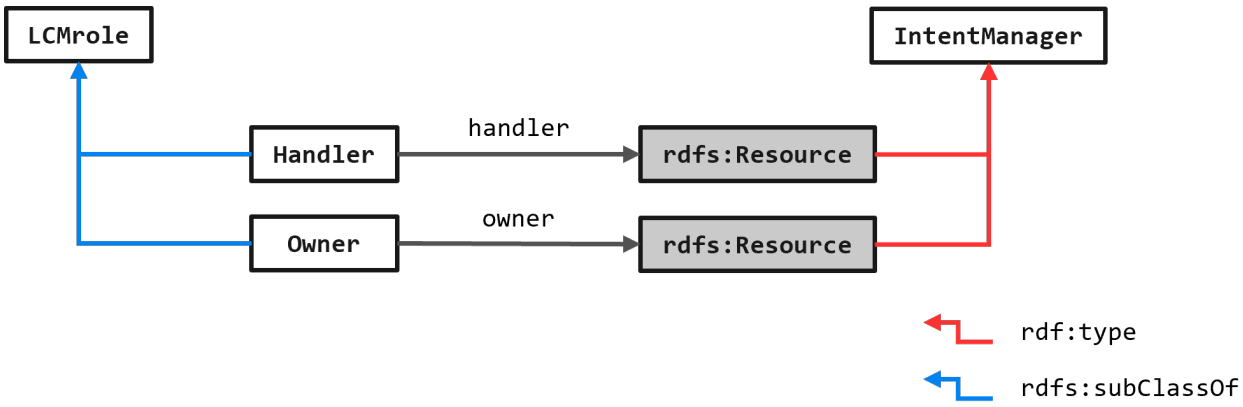

Figure 1: Roles of intent management functions

The classes `imo:Handler` and `imo:Owner` are instances of the class `imo:LCMrole`. The properties `imo:handler` and `imo:owner` allow referring to a resource that implements either of these roles. This resource is typically also an instance of the class `imo:IntentManager`. This structure establishes that there are two distinct types of intent management Functions and that they have the roles of owner and handler within the life cycle management of intent.

For example:

```
ex:Intent1
  rdf:type icm:Intent ;
  imo:owner ex:MarketplaceIMF ;

# ...

.
```

This example defines an intent as instance of class icm:Intent from the intent common model. It adds the information that the intent manager `ex:MarketplaceIMF` is in the role of intent owner for this intent.

3. Intent model classification

The TM Forum intent ontology consists of multiple distinct models and sub-ontologies. The intent common model is the mandatory core model of intent and intent report expression. It specifies the domain independent common vocabulary that every intent manager must support.

Intent extension models are optional extensions defining additional vocabulary to be used in intent expression. This can be extended, but still domain independent vocabulary. For example, TM Forum defines the intent validity model. It allows defining conditional validity of requirements within an intent.

A domain specific intent extension model would define vocabulary specific to use cases or subsystems. For example, an intent extension model for RAN management might specify the KPIs of a radio system to be used to define its required performance.

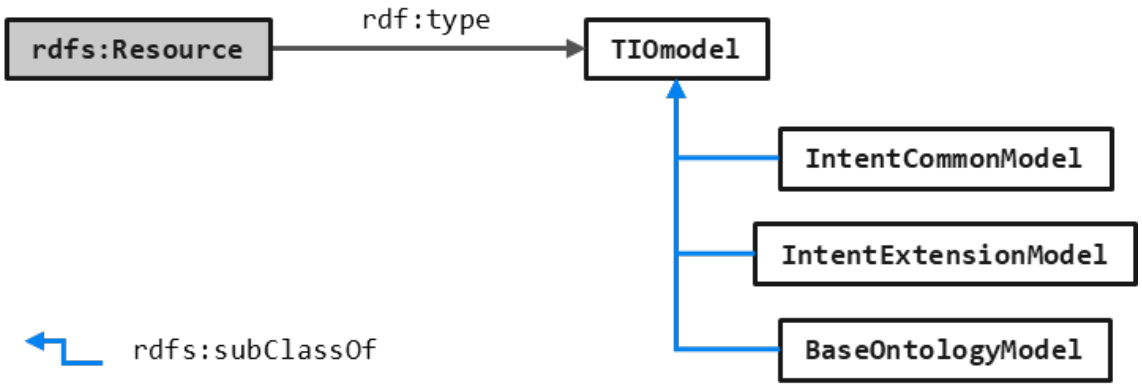

Figure 2: Categorization of models used in intent expression.

The class `imo:TIOmodel` expresses that the ontology model is part of the TM Forum intent Ontology. Its subclasses `imo:IntentCommonModel`, `imo:IntentExtensionModel` and `imo:BaseOntologyModel` express the role of the model within the TM Forum Intent Ontology.

An ontology of type `imo:IntentCommonModel` is a version of the TM Forum intent common model. An ontology of type `imo:IntentExtensionModel` is a version of an intent extension model. This is used in the definition of intent models, for example, the description of the intent common model can start like this:

```
<http://tio.models.tmforum.org/tio/v3.0.0/IntentCommonModel/>
  rdf:type imo:IntentCommonModel ;

# ...

.
```

This example defines that the resource identified by <http://tio.labs.tmforum.org/tio/v3.0.0/IntentCommonModel/> is an intent common model.

4. Associated values

Individuals in the TM Forum intent ontology can have an associated value. This value can be assigned directly using the `rdf:value` property. Functions used as property of an individual would provide an associated value as well.

Associated values are considered for classes which have the `tio:associatedValueType` property. It specifies the type the associated value of an instance of this class has. Allowed types are boolean as well as containers and quantities. For these data types a default value and a combination method are defined. Combination methods determine how the contributions from multiple contributors are handled in the individuals of a class that bears associated values.

Boolean: The default value is "false". Default combination method is logical conjunction equivalent to the `log:allOf` function.

Quantity: The default value is "0" with no unit. Default combination method is the sum of contributing values equivalent to the `quan:sum` function. Other numerical data types such as `xsd:integer`, `xsd:decimal` and `xsd:double` are covered as well due to implicit conversion into unitless quantities.

Container: The default value is an empty container. Default combination method is a union equivalent to the `set:union` function.

The datatype of the associated value of an individual of a class is defined using the `tio:associatedValueType` property.

The value combination method of an individual of a class is defined using the `tio:associatedValueCombination` property. It specifies the function to be used as combination operation. This function must have only arguments of the same type as the data type of the associated value and non-bounded arity. The evaluation of multiple contributors to the associated value would be done by using each contributor as argument to the function.

For example:

```
icm:Condition
  a rdfs:Class
  tio:associatedValueType xsd:boolean ;
.
```

This example is the definition of the class `icm:Condition` from the intent common model. A condition has an associated value of type `xsd:boolean`. A combination method is not specified; thus the default of logical conjunction applies.

For example:

```
ex:C1
  a icm:Condition
  quan:smaller ( [ met:lastValue ( ex:Latency ) ]
                 [ rdf:value "10"^^xsd:decimal ;
                   quan:unit "ms" ]
               )
  quan:greater ( [ met:lastValue ( ex:Throughput ) ]
                 [ rdf:value "100"^^xsd:decimal ;
                   quan:unit "Gbps" ]
               )
.
```

This example shows how two functions contribute to the boolean type associated value of the condition individual `ex:C1`. The default combination method of logical conjunction according to the `log:allOf function` applies. Assuming the condition expression involving latency is "true" and the condition expression regarding throughput is "false", the associated value of `ex:C1` is "false". An equivalent statement of the above would be:

```
ex:C1
  a icm:Condition
  rdf:value [ log:allOf ( [ quan:smaller ( [ met:lastValue ( ex:Latency ) ]
                                           [ rdf:value "10"^^xsd:decimal ;
                                             quan:unit "ms" ]
                                         )
                          ]
                          [ quan:greater ( [ met:lastValue ( ex:Throughput ) ]
                                           [ rdf:value "100"^^xsd:decimal ;
                                             quan:unit "Gbps" ]
                                         )
                          ]
                        )
            ]
.
```

This example shows an explicit assignment of an associated value. Thus is equivalent to the expression above. Here are the two contributions to the associated value are arguments of the `log:allOf` function. Its result value is then assigned as associated value to `ex:C1` explicitly using the `rdf:value` property.

5. Vocabulary of distinct models

The TM Forum Intent Ontology (TIO) is modular. It consists of constituent models addressing various aspects of intent based operation. The entire model for intent management is a federation of these constituent models specified as part of the TIO and might even contain models added from outside the TIO. However, many parts of the TIO are optional. Which models are fully or partially supported depends on the implementation of intent managers.

Intent managers need to communicate the models they support and vocabulary available to express intent. This is, for example, and fundamental building block of intent specifications [TR299] and will also be used in intent manager capability profiles added in a future release.

The central concept is model vocabulary. It refers to all classes, properties and object instances a distinct model specifies. For example, the intent common model [TR290] specifies the class intent referred to by the IRI/URI "`http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/Intent`". Often the first part "`http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/`" of this IRI is substituted with the prefix "`icm:`". This is done for better readability especially in TURTLE based expressions of the model. An equivalent short identification of the class intent is therefore "`icm:Intent`"

The first part of the IRI is the identification of the model. It is also referred to as the namespace of the mode. The last part of the IRI is vocabulary specified by the model. In this example, "Intent" is vocabulary specified within the intent common model with namespace "`http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/`".

Vocabulary of a distinct model are all artifacts that are specified under the model namespace. A model vocabulary is therefore a set of all these artifacts or more specifically a set of all the IRI/URI of them. The vocabulary supported by an intent manager is then also a set of all modeling artifacts this intent manager implements support for.

For facilitating a practical handling of vocabulary it is recommended that each distinct model at least specifies a container called "Vocabulary" within its namespace. The members of this container are all artifacts the model specifies represented by their IRI/URI. For example the vocabulary container `icm:Vocabulary` of the intent common model has the member `icm:Intent` next to `icm:Expectation`, `icm:Target` and all other elements the intent common model specifies.

Next to the container with complete vocabulary, it is also possible that models provide additional containers with partial vocabulary. This can be used, for example, to distinguish use cases.

6. Administrative Appendix

6.1. Document History

6.1.1. Version History

Version Number	Date Modified	Modified by:	Description of changes
1.0.0	07-Feb-2023	Alan Pope	Final edits prior to publication
3.0.0	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

6.1.2. Release History

Release Status	Date Modified	Modified by:	Description of changes
Pre-production	07-Feb-2023	Alan Pope	Initial release
Pre-production	17-Mar-2023	Adrienne Walcott	Updated to Member Evaluated status
Pre-production	11-Apr-2023	Alan Pope	Updated to v3.0.0
Pre-production	15-May-2023	Adrienne Walcott	Updated to Member Evaluated status
Pre-production	15-Aug-2023	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
Production	30-Aug-2024	Adrienne Walcott	Updated to reflect TM Forum Approved status

6.2. 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*

7. TR292A Intent Management Elements v3.6.0 - Appendix A Vocabulary

7.1. Appendix A: Vocabulary Reference

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

7.1.1. BaseOntologyModel

The class `imo:BaseOntologyModel` is a subclass of `imo:TIOmodel`. Its instances are models within the TM Forum Intent Ontology. Intent common models and intent extension models rely on the fundamental vocabulary and semantics defined by a base ontology model.

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

7.1.2. associatedValueCombination

The property `imo:associatedValueCombination` specifies the combination function to be applied if multiple contributors to the associated value of an instance a class are used.

Instance of: `rdf:Property`Domain: `rdfs:Class`Range: `fun:Function`

7.1.3. associatedValueType

The property `imo:associatedValueType` specifies the type of the associated value of instances of a class. If this property is used in the definition of a class, instances of this class have associated values.

Instance of: `rdf:Property`Domain: `rdfs:Class`

7.1.4. IntentCommonModel

The class `imo:IntentCommonModel` is a subclass of `imo:TIOmodel`. Its instances are intent common models within the TM Forum Intent Ontology.

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

7.1.5. IntentExtensionModel

The class `imo:IntentExtensionModel` is a subclass of `imo:TIOmodel`. Its instances are an intent extension model within the TM Forum Intent Ontology.

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

7.1.6. IntentManager

The class `imo:IntentManager` expresses that its instance is an Intent Management Function.

Instance of: `rdfs:Class`

7.1.7. LCMrole

The class `imo:LCMrole` expresses that its instance has a role in intent life-cycle management. Its subclasses specify which role that is.

Instance of: `rdfs:Class`

7.1.8. Handler

The class `imo:Handler` expresses that its instance has the role of intent handler in intent life-cycle management.

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

7.1.9. handler

The property `imo:handler` expresses that the resource of type `imo:IntentManager` in its object has the role of an intent handler for the subject. The subject is typically an intent or associated intent report.

Instance of: `rdf:Property`
Range: `imo:IntentManager`

7.1.10. owner

The property `imo:owner` expresses that the resource of type `imo:IntentManager` in its object has the role of an intent owner for the subject. The subject is typically an intent or associated intent report.

Instance of: `rdf:Property`
Range: `imo:IntentManager`

7.1.11. Owner

The class `imo:Owner` expresses that its instance has the role of intent owner in intent life-cycle management.

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

7.1.12. TIOmodel

The class `imo:TIOmodel` expresses that its instance is an ontology model within the TM Forum Intent Ontology

Instance of: `rdfs:Class`

7.1.13. Vocabulary

The object `imo:Vocabulary` Is a container of all model elements.

Instance of: `rdfs:Container`