TM Forum Introductory Guide

# Intent Manager Capability Profiles

IG1253D

Team Approved Date: 26-Nov-2021

| | |
|---|---|
| Release Status: Production | Approval Status: TM Forum Approved |
| Version 1.0.0 | 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](http://www.tmforum.org)

# Table of Contents

# List of Figures

# Executive Summary

In this document, we introduce a registration and discovery scheme involving an intent manager capability profile. Intent management functions can publish their capability profile including their scope of responsibilities through an intent manager registry function. The intent manager registry exposes a query interface allowing discovery of intent managers through the details of their profile.

This document introduces intent manager scopes, which express the operations tasks within distinct autonomous domains, system levels and sub-systems the intent manager is concerned with. The intent manager scope is part of the intent manager capability profile. We describe example profiles as based for partitioning the entirely of operational task and assigning subsets to distinct intent managers. This introduces modularity and specialization of intent managers.

Furthermore, we propose a model for expression of intent manager capability profiles and the interface exposed by the intent manager registry function.

# Introduction

Intent management functions were introduced in IG1253. They drive the intent life cycle, communicate intent and intent reports over the intent interface and coordinate autonomous operation within an autonomous domain, system layer or sub-system. The implementation of intent management functions can vary significantly with respect to capabilities as autonomous coordinators of intent based operation for a domain. Domains have various complexities being reflected in more or less feature rich intent management.

For example, a domain may employ a set of rule based policies for making operational decisions and for executing actions. This kind of implementation is typically not very adaptive to new situations, but for less complex domains with a stable and predictable environment this may still be the best choice. The need for updating rules to adapt the system is a human activity that would constitute a breach of autonomy. In stable environments this may only be needed in exceptional cases.

The intent manager in this example would not have access to sophisticated prediction models and therefore would not be able to support optional interface procedures such as JUDGE/PREFERENCE, PROBE/ESTIMATE and BEST/PROPOSAL. Other intent managers may have these capabilities as a result of a faster evolution of its implementation. The use of machine learning, probabilistic reasoning and digital twins can lead to better operational decisions with sophisticated conflict detection, risk analysis and prediction of action effects.

Intent managers can also differ significantly with respect to intent expressiveness. This manifests itself in different compositions of their supported model federation. This variation is naturally driven by the needs of the autonomous  domain and system the intent manager is associated with. Domain specific models in the federation deliver the domain vocabulary and semantics needed. On top of this intent extension functions also introduce further domain independent, but optional modeling artifacts. One example is the modeling of temporal validity of requirements as demonstrated in IG1253 B.

Following from this discussion it is a basic requirement that intent managers can detect which other intent mangers are available and what their scope of responsibility is. This determines if intent can be used to impact an operational domain with the intent paradigm of setting requirements.

Furthermore, an intent manager needs information about the capabilities of the intent manager within the targeted domain. This determines, for example, what expressiveness is available due to the supported model federation in the target intent manager. And it potentially limits the intent interface operations that can be used.

Intent Manager Scopes define the set of operational tasks and resources an intent manager is responsible for. This is a subset of all operational tasks and resources within an autonomous network. Every intent manager has a well-defined scope. It must not overlap with the scopes of other intent managers. This way direct conflicts in the actions of intent managers are avoided. Most importantly, intent manager scopes define which intent manager is responsible to handle a particular requirement. This determines how requirements need to be distributed to distinct intents and where these intents need to be sent to. Intent manager scopes are therefore mandatory information within an intent manager profile.

This document proposes a model for expressing Intent Manager Capability Profiles. The model presented here is based on the Resource Description Framework (RDF) [rdf] family of models. It was chosen for consistency with the other models presented in the IG1253x suit of documents. And it is also using base ontologies, such as the intent management ontology introduced in IG1253 and the intent interface ontology described in IG1253C. They provide a convenient base vocabulary. However, the considerations that made RDF/RDFS the modelling standard of choice for intent and intent report models do not necessarily apply to the modeling of intent manager capability profiles or to the related interface models for publication and discovery. Other model families, such as UML appear perfectly well suited for these purposes.

# 1. Intent Management Scopes

Each instance of an intent management function has a well-defined scope of operation. This usually corresponds to an autonomous domain and level in the technical architecture or a subsystem within the autonomous network. Practically it refers to a set of resources and a corresponding set of operational tasks and processes an individual intent manager is responsible for. This responsibility implies that it is the only intent manager authorized to use these resources for operation and it is the only one making or coordinating the decisions and actions concerning resources and functions within its scope. Consequently, this intent manager has the sole responsibility to fulfil the intent targeting this scope.

The operational scope of and intent management function is referred to as an Intent Management Scope or intent manager scope. Both terms are used synonymously.

Intent managers are not allowed to directly interfere and interact with the intent operation that is in scope of another intent manager. An intent manager can however use intent to put additional requirements on the operation of another intent manager scope, asking the respective other intent manager reach a state that fulfills all needs. Intent management functions are in this respect single points of contact for all operational request within the intent mechanism.

In this environment it is therefore essential that intent management functions know each other's scope of responsibility. This allows them to avoid conflicting overlaps in decisions and actions and it also allows to identify the right destination for a requirement and package it into intent accordingly. This means an intent manager wanting to put requirements, would know which other intent manager is responsible for handling it due to its expressed scope. In this respect intent management scopes partition the entire intent aware autonomous operation and uniquely assign responsibilities to the intent managers within.

Every intent manager scope corresponds to exactly one instance of the intent management function. An autonomous network typically consists of multiple intent manager scopes and all of them together define the vertical and horizontal range of intent-driven operation. The horizontal range refers to the number of operational domains on the same operational layer that are automated through intent driven operation concepts. Examples are autonomous domains in resource operation, such as cloud, transport and RAN management. Each of them would employ a dedicated intent manager with capabilities and an implementation that matches the needs of that domain.

The vertical range refers to further layering into sub-domains through nested intent manager scopes. This is achieved by splitting responsibilities into lower and higher level tasks and implement separate intent management functions. The higher level intent manager can then use intent to influence the lower level operation by defining its operational requirements and goals and form an inter-layer control loop that closes through intent reporting. This means Intent manager scopes can be nested creating a hierarchy and subdivision into smaller scopes.

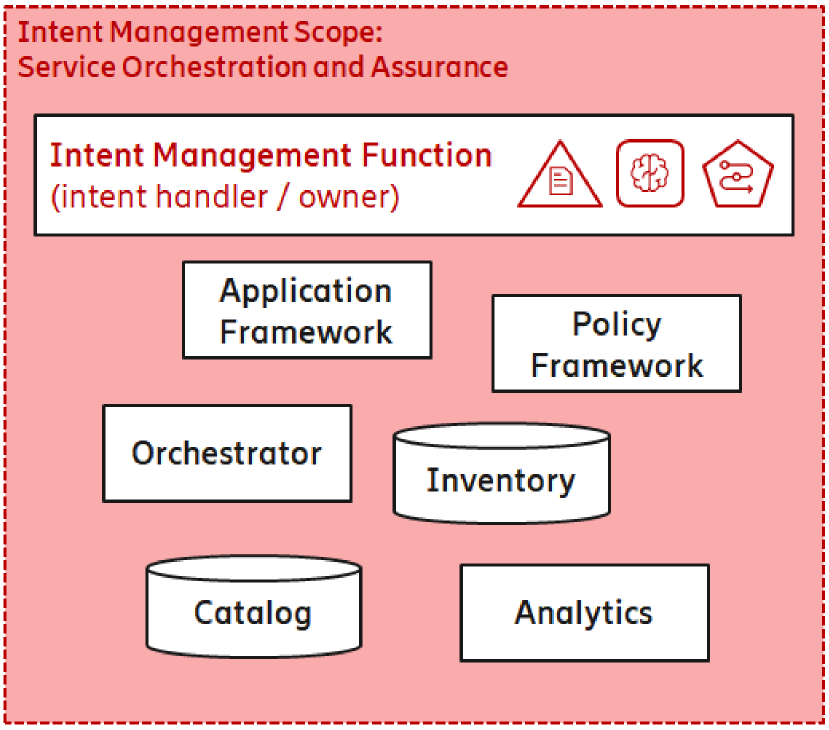

**Figure 1.1: Example intent management scope**

Figure 1.1 shows an example of an intent manager scope. In this example, the intent handling scope of service orchestration and assurance is shown. This is an example scope from the service operation layer in the autonomous network architecture [ig1230]. It refers to intents being used to communicate requirements of services to be delivered.

This example intent manager scope contains one intent management function. It receives service-related intents, and it tries to comply to the intent utilizing and collaborate with other functions of the service operation layer. Applications and policies determine the decisions that can be made and processes that are available within this scope. An orchestrator function would be able to choose and allocate resources needed for realizing the service. There are catalogs of available service elements, inventories containing the currently instantiated resources and analytics functions that help to determine and interpret the system state. These functions are typical for service operation. Other scopes would consist of one intent manager in combination with a number of functions and services specific to the domain and level.

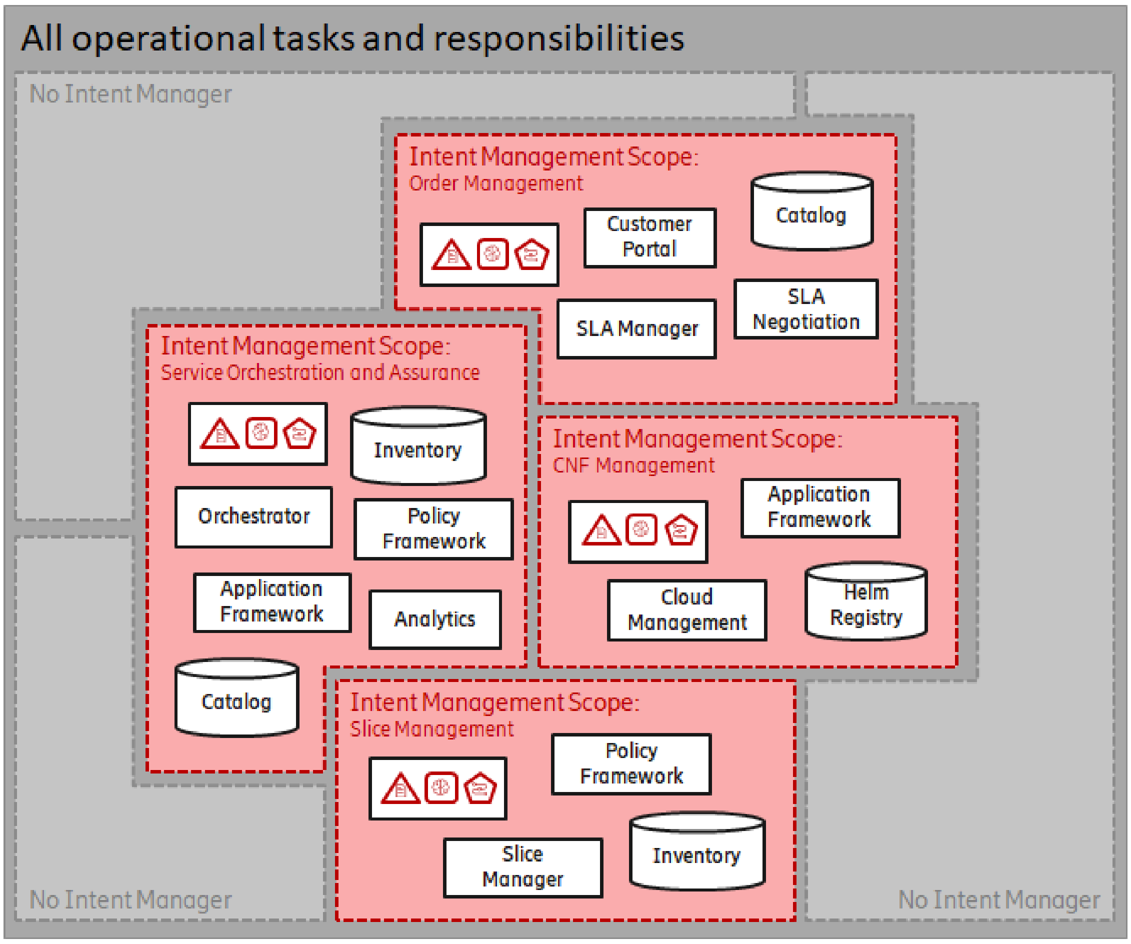

**Figure 1.2: Multiple example intent management scopes**

Figure 1.2 shows several intent manager scopes. Each of them contains exactly one instance of an intent management function implemented specifically implemented for the responsibilities needed in its scope. Other management functions of the autonomous network are considered to be part of this scope if the function and service they provide are relevant to the tasks for which the particular intent management function is responsible. For example, the intent manager scope of order management would include functions for SLA negotiation, a catalog of available service products, and portals that implement front-ends for customer engagement.

Typically, the entire operating system is not aligned to the intent paradigm. There are scopes of responsibility within the operating system that are not covered by intent management. This means that the respective autonomous domain or system has not yet been converted to intent-based operation yet or the nature of the tasks does not allow operation through intent management.

The intent manager scope is a key entry in the intent manager capability profile, which describes an instance of the intent management function. It summarizes and publishes responsibilities and range of operation. Typically, the intent manager scope implies certain expressiveness and content that can be used in the intent targeting it.

## 1.1. Governance of Intent Management Scopes

Intent management scopes should be defined collaboratively by multiple SDOs. SDOs and workgroups within SDOs have specific competences and mission statements with well-defined scopes. We expect that this naturally matches a set of intent handling scopes. This also implies that the respective work-group within an SDO would define the capabilities of an intent manager within this scope. This includes a proposal, which models the intent manager shall support and it will also define and name the intent manager responsibility scope. This means central governance of intent management scopes is not strictly needed, and we expect to see them being proposed from multiple parties and organizations.

## 1.2. Collection of Intent Management Scopes

This chapter defines scopes of intent managers. For the time being this collection of intent manager scopes is provided to exemplify the principle. The list is not meant to be complete or final. We expect that a practical list of intent handling scopes would be defined over time. This would involve future studies of use-cases. It would also involve multiple SDOs, because working groups with an agenda to study and standardize a particular autonomous domain can best decide a sensible partitioning of intent handling. In this respect the definition of intent handling scopes is directly related to the definition of domain specific intent extension models.

The following sub-chapters present example intent manager scopes within the major operational layers of an autonomous network.

### 1.2.1. In Business Operation Exemplar

**Contract Negotiation Scope**

This is the scope of an intent manager that coordinates automated contract negotiation. All intent that expresses requirements of the operator with respect to acceptance of contracts from financial and marketing perspective would be in scope.

**Order Management Scope**

This is the scope of an intent manager that receives customer orders for service products and creates intent requiring their fulfillment and assurance from the underlying system.

**Regulatory Policies Scope**

This scope refer to an intent manager that translates regulatory and legal requirements into intent. This way local market legislation and regulation can be considered within the intent based autonomous system.

**Operator Policies Scope**

This is the scope of an intent manager that allows the operator to formulate strategic business policies and issue them as intent for the autonomous system to consider.

### 1.2.2. In Service Operation Exemplar

**Service Orchestration and Assurance**

The intent manager of this scope receives all intent about services that need to be delivered. It interacts with orchestration and assurance systems to break down the service intent resources to be used and their configuration across multiple autonomous domains. Typically, this intent manager would act by setting multiple intents in resource operation.

### 1.2.3. In Resource Operation Exemplar

**Core Network Intent Management Scope**

This is the scope of an intent manager that has the responsibility for core network resources and the related operational tasks.

**Transport Management Scope**

This is the scope of an intent manager that has the responsibility for transport resources and the related operational tasks.

**Slice Management Scope**

This is the scope of an intent manager that has the responsibility for setup and assurance of slices.

**Network Function Management Scope**

This is the scope of an intent manager responsible to coordinate the deployment and assurance of network functions.

# 2. Intent Manager Capability

Intent managers need to be aware of other intent managers' presence in the system as well as their capabilities. For example, intent managers in the role of intent owners decide to use intent for putting requirements on the operation of another autonomous domain. First this requires to detect if the targeted domain is participating in the intent mechanism by exposing an intent management function. To send an intent to it is also must be capable to assume the role of intent handler for that domain. The intent manager scope is the key information determining responsibility of intent managers. Knowledge about the address of the intent manager would then allow to actually contact it over the intent interface and send the intent.

All intent managers have to support the mandatory procedures of the intent interface. Other intent managers need to know if optional procedures are also supported and can be used in the communication with an intent manager. For example, and intent owner might benefit from the availability of PROBE/ESTIMATE or BEST/PROPOSAL procedures when assessing the feasibility of requirement details it wants to put into an intent. Intent handlers should know if the owner of an intent supports the JUDGE/PREFERENCE procedure allowing asking questions back to the owner for optimizing operational decision. The support of interface procedures can be further differentiated with respect to supported versions.

A central aspect of an intent manger's capability is its support for models within a model federation for intent and intent report expression. This is typically a federation of domain independent generic models such as the intent common model or RDF/RDFS as modelling base, and any number of domain dependent intent extension and intent information models. IG1253A defines the intent common model and IG1253B proposes and defined several intent extension models.

The model defined in Appendix A of this document provides vocabulary to formulate intent manager capability profiles. Chapter 3 introduces the intent manager registry. It provides interfaces for publishing intent manager capability profiles and for discovery of intent managers through its published profile information.

# 3. Intent Manager Registry

This document proposes the introduction of an intent manager registry function. It stores information about all available intent managers. Furthermore, it enables the discovery of intent managers through queries about the profiles.

The interfaces of the intent manager registry are shown in Figure 3. The intent manager registration interface allows uploading intent manager capability profiles to the intent manager registry. This publishes the presence of an intent manager and its scope and capabilities. The intent manager discovery interface allows searches in the published intent manager capability profiles.

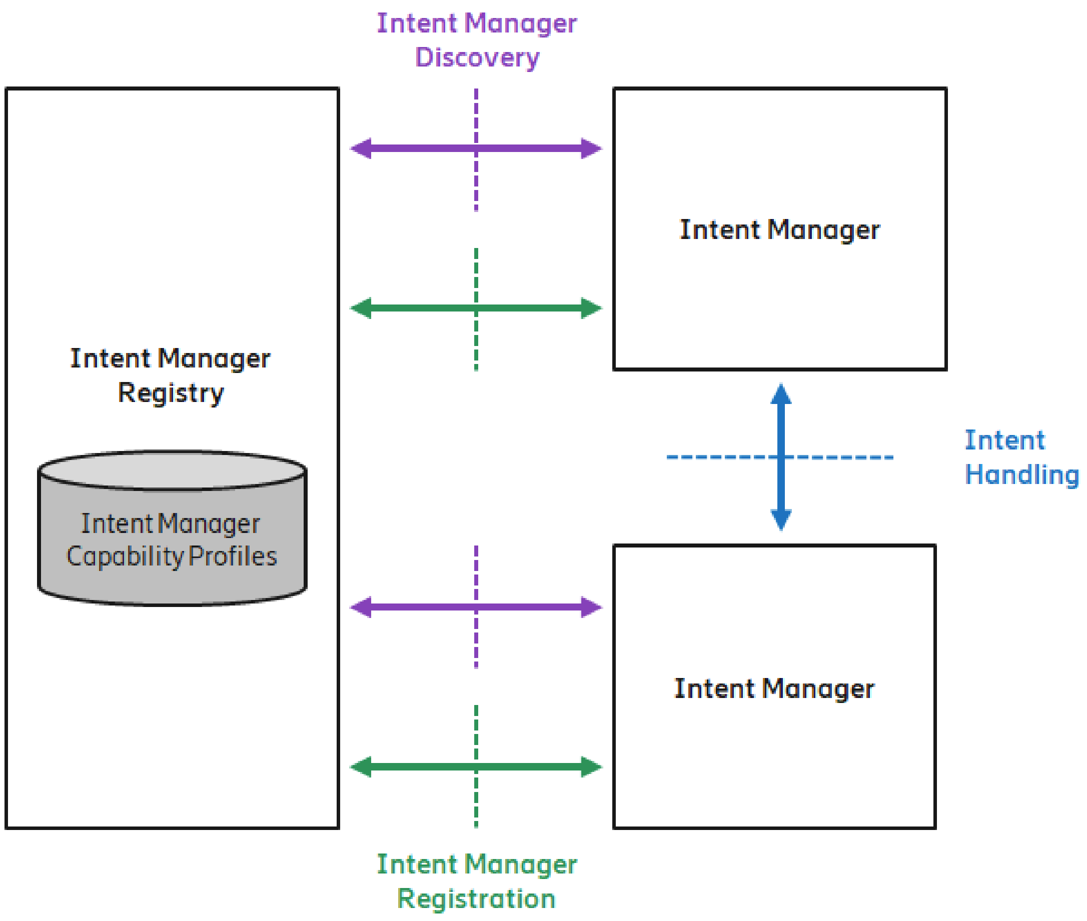

**Figure 3.1: Intent manager registration and discovery**

Intent manager discovery is essential for intent owners deciding if they can use intent towards an autonomous domain and what they can express with this intent. The discovery answers the question if there is a respective intent manager present in the system with a scope that matches the target domain. On the other hand intent handlers need discovery to get access to the capabilities of the intent owner. This shows for example what vocabulary can be used in intent reports and if the JUDGE/PREFERENCE procedure is supported by the owner.

## 3.1. Registration of intent managers

An autonomous network consists of multiple instance of intent management function. Each of them is having a unique implementation that matches the autonomous domain, operational layer and sub-system it is responsible for. This means every intent manager has unique set of capabilities as well as a defined scope of responsibilities. To make this information available to other functions, and intent manager creates a profile about its own scope and capabilities. It then sends the profile to the intent manager registry.

Depending on the implementation of an intent manager, its capability profile might not be constant. It can change, for example, if policies or apps are added or machine learned models are replaced with an improved version. In modern systems these are artifacts with own life cycles and changes in them can introduce or remove features with an effect on capabilities of an intent manager.

The details of the interface deign are future work. Capability publication interfaces are a common feature in adaptable systems. Therefore, standard methods and processes for designing the respective interface registration and publication interface and be applied.

## 3.2. Discovery of intent manager

The intent manager registry exposes an interface that allows to query intent manager capability profiles. In this document we are using models based on RDF/RDFS for expressing the intent manager capability profile. Therefore, it would be sensible to base the discovery interface on a query language designed and standardized for RDF. The query language proposed to be used for discovery of intent managers is SPARQL [sparql]. SPARQL is a standardized query language inspired by SQL style database queries. It adds expressiveness for queries about ontology based knowledge graphs. Many graph databases support SPARQL as their main query language or as option.

# 4. Appendix A: Intent Manager Capability Profile Model

## 4.1. Motivation and background

Intent management functions need to publish their capability profile. The intent manager registry function is the receiver of this profile and it adds into its knowledge base. This chapter proposes a model providing the vocabulary to define the intent manager capability profiles.

## 4.2. Notation and namespaces

| Model | Prefix | Namespace | Published by |
|---|---|---|---|
| Intent Manager Capability Profile | imcp | https://www.tmforum.org/2020/07/IntentManagerCapabilityProfile * | TM Forum |
| Intent Management Ontology | imo | https://www.tmforum.org/2020/07/IntentManagmentOntology * | TM Forum |
| Intent Interface Ontology | iio | https://www.tmforum.org/2020/07/IntentInterfaceOntology * | TM Forum |
| W3C RDF version 1.1 | rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# | W3C |
| W3C RDF Schema 1.1 | rdfs | http://www.w3.org/2000/01/rdf-schema# | W3C |
| XML Schema | xsd | http://www.w3.org/2001/XMLSchema# | W3C |
| Example namespace | ex | http://example.com/IntentModeling | n/a |

*: Proposed IRI to show the concept. It might be different when the model is published.

This model is referred to Intent Manager Capability Profile and "imcp:" is used as namespace prefix in this document.

The model is based on the RDF/RDFS modelling family. It uses the general ontology of intent management as well as the intent interface ontology.

## 4.3. Principles and vocabulary overview

Intent management functions differ with respect to the domain they manage, but also regarding their capabilities. They might support a different set of intent extension models, serialization formats and intent interface procedures. This model defines vocabulary for expressing these aspects of an intent manager capability and thus allow to formulate an intent manager capability profile.

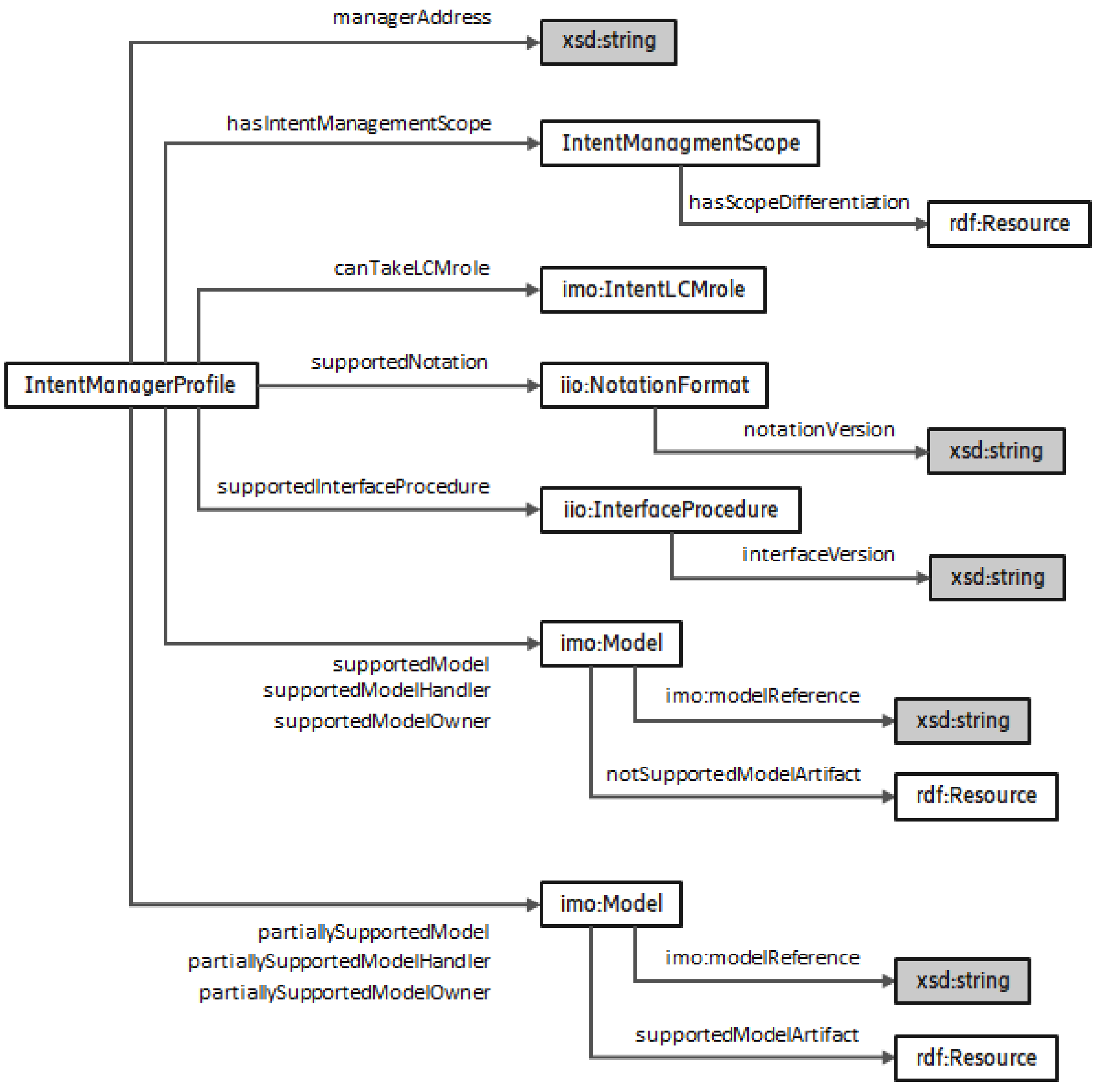

**Figure 4.1: Overview of classes and properties**

### 4.3.1. Intent manager address and contact information

Allows specifying how to contact the instance of the intent management function the profile is about. This is used for addressing of messages on the intent interface towards this instance. The string can for example contain the IRI that is representing the intent manager instance.

### 4.3.2. Intent Management Scope

This property allows to define the intent management scopes this intent manager is responsible for. Usually there is one scope per intent manager, but it is allowed to define multiple scopes if the responsibility of this instance of the intent manager extents to all of them.

The scope Identifier usually implies a domain specific implementation of the intent manager. Further differentiation of scopes is possible if the domain is further partition with multiple instances of intent management. This can for example be geographic partitioning if the network is divided into separately operated regions. The exact vocabulary for different types and dimensions of scope differentiation will be studied through use cases and operator feedback. It is left undefined in the model for now.

### 4.3.3. Intent LSM role

An intent manager has the capability to take the role of an intent owner, intent handler or both. An intent handler that only takes the owner role usually a top level source of intent. It would be associated with user portals or management functions that define requirements, but are themselves not intent aware. This management function does not implement the role of intent handler and it states this in its capability profile.

### 4.3.4. Supported notation format

Intent and intent reports are knowledge graphs in the form of an ontology. For sending them over the intent interface these graphs need to be serialized. Multiple notation formats are available for this, for example TURTLE, RDF/XML or JSON-LD. The intent manager states its support for notation formats in its capability profile. It can further differentiate by supported version of the notation format.

### 4.3.5. Supported interface procedures

The intent interface defines four sets of procedures:

- The SET/REMOVE/REPORT interface procedure. One version of this procedure is mandatory to support.
- The JUDGE/PREFERENCE interface procedure
- The PROBE/ESTIMATE interface procedure
- The BEST/PROPOSAL interface procedure

The intent manager capability model allows to specify which of them is supported. Options are defined in the intent interface ontology [ig1253c]. Further procedures might be introduced later and would then be reflected here.

The model allows specifying which versions of the interface procedures are supported. In this respect it is sensible to also specify the mandatory SET/REMOVE/REPORT procedure and state which exact version are supported.

### 4.3.6. Supported Models

This defines the models that can be used in intent and intent reports and would be understood by the intent manager. This implicitly defines the model federation that is possible to use with the intent manger.

It is possible to define models that are considered fully supported with the imp:supportedModels property and then define exceptions with the imp:notSupportedModelArtifact property.

Alternatively it is possible to mention a model as partially supported with the imp:partiallySupportedModels property. This requires that all model artifacts that are supported must be explicitly mentioned with the imp:supportedModelArtifact property.

It is further possible to state separately if a model and its artifacts are supported by the intent manger when it has the role of intent owner or intent handler. This means some models are only supported in intents that are received and other only in intents that the intent manager creates and sends to other intent managers for handling.

Model artifacts that can be explicitly supported or not refers to the entire vocabulary specified by the model. Typical examples are classes, properties and individuals defined in the model.

## 4.4. Vocabulary specification

This chapter defines the classes, instances and properties of the intent manager capability model.

Please note that the formulation of capability profiles uses vocabulary defined in these models in combination with definitions from the intent management ontology [ig1253] and the intent interface ontology [ig1253C].

### 4.4.1. Classes

| **Class:** | **imcp:IntentManagerProfile** |
|---|---|
| Definition: | An object of class imp:IntentManagerProfile represents the capability description profile of intent management function |
| Instance of: | rdfs:Class |

| **Class:** | **imcp:IntentManagemetScope** |
|---|---|
| Definition: | Describes the responsibility scope of an intent management function |
| Instance of: | rdfs:Class |

### 4.4.2. Instances

The following instances of imcp:IntentManagmentScope are defined. This list is not final or complete. It demonstrates how intent manager responsibility scopes can be enumerated

| **State Individual** | **Description** |
|---|---|
| imcp:ScopeSliceManagment | The scope of an intent management function that is responsible for slice management |
| imcp:ScopeTransport | The scope of an intent management function that is responsible for transport management |

| State Individual | Description |
|---|---|
| imcp:ScopeRadio | The scope of an intent management function that is responsible for radio managment |
| imcp:ScopeContractNegotiation | The scope of an intent management function that is responsible for automated negotiation of contracts and SLA |
| imcp:ScopeOrderManagement | The scope of an intent management function that is responsible for automated ordering |
| imcp:ScopeRegulatoryPolicies | The scope of an intent management function that is responsible for translating regulatory and legal requirements into intent |
| imcp:ScopeOperatorPolicies | The scope of an intent management function that is responsible for translating operator business policies into intent |
| imcp:ScopeService | The scope of an intent managment function that is responsible for service orchestration and assurance |
| ... | ... |

### 4.4.3. Properties

| Property: | imcp:canTakeLCMrole |
|---|---|
| Definition: | Refers to a model that issupported |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |
| Range: | imo:intentLCMrole |

| Property: | imcp:hasIntentManagerProfile |
|---|---|
| Definition: | The property icm:ownerAddress allows to assign the IRI/URI of an intent owner to an icm:Owner object. This specifies the address that can be used for sending intent reports |
| Instance of: | rdf:property |
| Domain: | imo:IntentManager |
| Range: | imcp:IntentManagerProfile |

| Property: | imcp:hasIntentManagmentScope |
|---|---|
| Definition: | Refers to a model that issupported |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |
| Range: | imcp:IntentManagementScope |

| **Property:** | **imcp:hasScopeDifferentiation** |
|---|---|
| Definition: | allows assigning further differentiation of scopes. |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagement |
| Range: | rdf:Resource |

| **Property:** | **imcp:interfaceVersion** |
|---|---|
| Definition: | Specifies the version of an interface procedure |
| Instance of: | rdf:property |
| Domain: | imcp:InterfaceProcedure |
| Range: | imcp:IntentExtensionModel |

| **Property:** | **imcp:managerAddress** |
|---|---|
| Definition: | The property imcp:managerAddress allows to assign the IRI/URI of an intent owner to an icm:Owner object. This specifies the address that can be used for sending intent reports |
| Instance of: | rdf:property |
| Domain: | imo:IntentManager |
| Range: | xsd:string |

| **Property:** | **imcp:notSupportedModelArtifact** |
|---|---|
| Definition: | Defines artifacts and objects specified within the models that are explicitly not supported by the intent manager. |
| Instance of: | rdf:property |
| Domain: | imcp:IntentModel |
| Range: | rdf:Resource |

| **Property:** | **imcp:partiallySupportedModel** |
|---|---|
| Definition: | Refers to a model that is supported for intent and intent report expression.<br>The models are supported in handler as well as owner roles.<br>Artifacts in the mentioned model are assumed to be not supported unless explicitly referred to using the imcp:supportedModelArtifact property. |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |
| Range: | imo:IntentModel |

| **Property:** | **imcp:partiallySupportedModelHander** |
|---|---|
| Definition: | Refers to a model that is supported for intent and intent report expression.<br>This property means that these models can be used in intent that is received from other intent managers, but they are not necessarily supported in intent that is owned and send to others.<br>Artifacts in the mentioned model are assumed to be not supported unless explicitly referred to using the imcp:supportedModelArtifact property. |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |
| Range: | imo:IntentModel |

| **Property:** | **imcp:partiallySupportedModelOwner** |
|---|---|
| Definition: | Refers to a model that is supported for intent and intent report expression.<br>This property specifies that these models can be used in intent that is sent to other intent managers for handling them, but they are not necessarily supported in received intent.<br>Artifacts in the mentioned model are assumed to be not supported unless explicitly referred to using the imcp:supportedModelArtifact property. |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |
| Range: | imo:IntentModel |

| **Property:** | **imcp:supportedInterfaceProcedure** |
|---|---|
| Definition: | Refers to a model that is supported for intent and intent report expression |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |
| Range: | iio:InterfaceProcedure |

| **Property:** | **imcp:supportedModel** |
|---|---|
| Definition: | Refers to a model that is supported for intent and intent report expression.<br>The models are supported in handler as well as owner roles.<br>The mentioned model is assumed to be fully supported. Exceptions can be defined using the imp:notSupportedModelArtifact property. |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |

| **Property:** | **imcp:supportedModel** |
|---|---|
| Range: | imo:IntentModel |

| **Property:** | **imcp:supportedModelHandler** |
|---|---|
| Definition: | Refers to a model that is supported for intent and intent report expression.<br>This property means that these models can be used in intent that is received from other intent managers, but they are not necessarily supported in intent that is owned and send to others.<br>The mentioned model is assumed to be fully supported. Exceptions can be defined using the imp:notSupportedModelArtifact property. |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |
| Range: | imo:IntentModel |

| **Property:** | **imcp:supportedModelOwner** |
|---|---|
| Definition: | Refers to a model that is supported for intent and intent report expression in the role of intent Owner.<br>This property means that these models can be used in intent that is sent to other intent managers for handling them, but they are not necessarily supported in received intent.<br>The mentioned model is assumed to be fully supported. Exceptions can be defined using the imp:notSupportedModelArtifact property. |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |
| Range: | imo:IntentModel |

| **Property:** | **imcp:supportedModelArtifact** |
|---|---|
| Definition: | Defines, which artifacts in models that are explicitly supported by the intent manager |
| Instance of: | rdf:property |
| Domain: | imcp:IntentModel |
| Range: | rdf:Resource |

| **Property:** | **imcp:supportedNotation** |
|---|---|
| Definition: | Refers to a notation format that is supported to be used for serializing the intent and intent report graphs |
| Instance of: | rdf:property |
| Domain: | imcp:IntentManagerProfile |
| Range: | iio:NotationFormat |

## 4.5. Model usage and examples

### 4.5.1. Typical Intent Manager Capability Profile

This example demonstrates a typical intent manager capability profile

```
ex:ExampleIntentManagerProfileXYZ
  a imp:IntentManagerProfile ;


  imcp:managerAddress
"https://www.operator.com/AutonomousNetwork/Core/SliceManagment/IntentMan
ager/" ;

  imcp:hasIntentManagementScope
    [ a imcp:ScopeSliceManagment ;
      imcp:hasScopeDifferentiation
        [ ope:intentManagerRegion ope:NetworkRegionNorthEast ,
                                  ope:NetworkRegionNorthWest
        ] ;
    ] ;

  imcp:canTakeLCMrole imo:intentOwner, imo:intentHandler ;

  imcp:supportedNotation
    [ a iio:TURTLEnotation ;
      imcp:notationVersion "1.0" ;
    ] ,
    [ iio:RDFXMLnotation ,
      imcp:notationVersion "1.0" ;
      imcp:notationVersion "1.0" ;
    ] ;

  imcp:supportedInterfaceProcedure
    [ a iio:SetProcedure ;
      imcp:interfaceVersion "1.0" ;
      imcp:interfaceVersion "1.1" ;
    ],
    [ a iio:JudgeProcedure ;
      imcp:interfaceVersion "1.0" ;
    ],
    [ a iio:BestProcedure ;
      imcp:interfaceVersion "1.0" ;
    ] ;

  imcp:supportedModels
    [ a imo:Model ;
      imo:modelReference "http://www.w3.org/1999/02/22-rdf-syntax-ns#" ;
    ] ,
    [ a imo:Model ;
      imo:modelReference "http://www.w3.org/2000/01/rdf-schema#" ;
    ] ,
    [ a imo:IntentCommonModel ;
      imo:modelReference
"https://www.tmforum.org/2020/07/IntentCommonModel" ;
    ] ,
    [ a imo:IntentCommonModel ;
      imo:modelReference
"https://www.tmforum.org/2021/12/IntentCommonModel" ;
    ] ,
    [ a imo:IntentExtensionModel ;
      imo:modelReference
"https://tmforum.org/IntentExtension/2021/12/IntentValidity" ;
    ] ,
    [ a imo:IntentExtensionModel ;
      imo:modelReference
"https://tmforum.org/IntentExtension/2021/12/IntentTemporalValidity" ;
    ] ,
    [ a imo:IntentExtensionModel ;
      imo:modelReference "http://www.w3.org/2006/time" ;
    ] ,
    [ a imo:IntentExtensionModel ;
      imo:modelReference
"https://www.sdo2.com/SliceManagementIntent/2021/12/" ;
      imcp:notSupportedModelArtifact
"http://www.sdo2.com/SliceManagementIntent/2021/12/SliceLatency" ,

"http://www.sdo2.com/SliceManagementIntent/2021/12/SliceThroughput" ,
    ] ;

  imcp:partiallySupportedModels
    [ a imo:IntentExtensionModel ;
      imo:modelReference "http://www.sdo1.org/Metrics/Version2" ;
      imcp:supportedModelArtifact
"http://www.sdo1.org/Metrics/Version2/UserExperienceKPI/Latency" ,

"http://www.sdo1.org/Metrics/Version2/UserExperienceKPI/Throughput" ,

"http://www.sdo1.org/Metrics/Version2/SliceKPI/Latency" ,

"http://www.sdo1.org/Metrics/Version2/SliceKPI/Troughput" ;
    ] .
```

This intent manager capability profile example defines the capability details of an instance of an intent management function that can be reached through the URI specified by the imp:managerAddress property. This is the URI used as destination address for all operations on intent interface.

The intent manager capability profile then defines the responsibility scope of this intent manager. In this example the intent manager has the responsibility for intent about slice management.

This instance of the intent management function states that it can take the roles of intent owner as well as intent handler.

It supports intent and intent reports being transmitted using TURTLE, RDF/XML and JSON as notation format of the intent and intent report ontology graphs.

This intent manger supports the optional interface procedures JUDGE/PREFERENCE as well as BEST/PROPOSAL.

The intent manager capability profile then uses the imp:supportedModel and imp:partiallySupportedModel properties to list all models that can be used to formulate intent and intent reports so that the intent manager understands the vocabulary and semantics. In this respect imp:supportedModel specifies that listed model is fully supported unless exceptions are explicitly specified. This is done for the Slice Management Intent model by SDO2. The model is fully supported except Slice Latency and Slice Throughput KPIs.

This intent manager supports RDF and RDFS as the base models. It also supports an intent common model. The capability profile contains two distinct intent common models. These are different versions of the TM Forum intent common model.

The imcp:partiallySupportedModels property list models that are by default not supported and all supported artifacts defined in these models needs to be explicitly listed using an imcp:supportedModelArtifact property. In this example the model defined by SDO1 is a collection of KPI. From all defined KPI only the four explicitly mentioned are supported by this intent manager.

# 5. Appendix B: Abbreviations and acronyms

| | |
|---|---|
| ANF | Autonomous Networks Framework |
| BSS | Business Support System |
| DSL | Domain Specific Language |
| IETF | Internet Engineering Task Force |
| IoT | Internet of Things |
| IRI | Internationalized Resource Identifier |
| ISO | International Organization for Standardization |
| JSON | JavaScript Object Notation |
| JSON-LD | JavaScript Object Notation for Linked Data |
| KPI | Key Performance Indicator |
| MnF | Management Function |
| MnS | Management Service |
| MOF | Meta Object Facility |
| OCL | Object Constraints Language |
| OMG | Object Management Group |
| OWL | Web Ontology Language |
| RDF | Resource Description Framework |
| RDFS | RDF Schema |
| RACI | Responsible, Accountable, Consulted, Informed |
| RAN | Radio Access Network |
| SDO | Standards Defining Organization |
| SKOS | Simple Knowledge Organization System |
| SHACL | Shapes Constraint Language |
| SHEX | Shape Expression Language |
| SPARQL | Protocol And RDF Query Language |
| SQL | Structured Query Language |
| TURTLE | Terse RDF Triple Language |
| URI | Uniform Resource Identifier |
| W3C | World Wide Web Consortium |
| XMI | Metadata Interchange |
| XML | eXtensible Markup Language |
| YAML | Yet Another Markup Language |

# 6. Appendix C: References

[ig1235A] Intent Modeling v1.1.0

[ig1235B] Intent Extension and Information models v1.0.0

[ig1235C] Intent Life cycle management and Interface v.1.1.0

[ig1235D] Intent Manager Scope and Capability Management v1.0.0

[jsonld] JSON-LD 1.1, W3C Recommendation, 16 July 2020, https://www.w3.org/TR/2020/REC-json-ld11-20200716/

[rdf] RDF 1.1 Concepts and Abstract Syntax, W3C, https://www.w3.org/TR/rdf11-concepts/

[rdfjson] RDF 1.1 JSON Alternate Serialization (RDF/JSON), W3C Working Group Note, 07 November 2013, https://www.w3.org/TR/2013/NOTE-rdf-json-20131107/

[rdfprim] W3C, RDF Primer, https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/

[rdfs] W3C, RDF Schema 1.1, W3C Recommendation 25 February 2014, https://www.w3.org/TR/2014/REC-rdf-schema-20140225/

[rdfxml] RDF 1.1 XML Syntax, W3C Recommendation, 25 February 2014, https://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/

[sparql] SPARQL 1.1 Overview, W3C Recommendation 21 March 2013, https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/

[turtle] W3C, RDF 1.1 Turtle, Terse RDF Triple Language, W3C Recommendation, 25 February 2014, https://www.w3.org/TR/2014/REC-turtle-20140225/

# 7. Administrative Appendix

## 7.1. Document History

### 7.1.1. Version History

| Version Number | Date Modified | Modified by: | Description of changes |
|---|---|---|---|
| 1.0.0 | 26-Nov-2021 | Alan Pope | Initial Release |

### 7.1.2. Release History

| Release Status | Date Modified | Modified by: | Description of changes |
|---|---|---|---|
| Pre-production | 26-Nov-2021 | Alan Pope | Final edits prior to publication |
| Production | 21-Jan-2022 | Adrienne Walcott | Updated to reflect TM Forum Approved Status |

## 7.2. Acknowledgments

### 7.2.1. Guide Lead & Author

| Member | Title | Company |
|---|---|---|
| Jörg Niemöller | Expert of Analytics and Customer Experience | Ericsson |

### 7.2.2. Main Contributors

| Member | Title | Company |
|---|---|---|
| Jörg Niemöller | Expert of Analytics and Customer Experience | Ericsson |
| Kevin McDonnell | Senior Director, Intelligent Automation | Huawei |
| James O'Sullivan | Product Director, Intelligent Automation | Huawei |
| Dave Milham | Chief Architect | TM Forum |
| Vinay Devadatta | Practice Head (Innovation & Industry Relations) | Wipro Technologies |
| Azahar Machwe | OSS Automation | BT Group plc |
| Wang Lei | Systems Expert | Huawei |
| Tayeb Ben Meriem | Senior Standardization Manager (OSS) | Orange |
| Leonid Mokrushin | Principle Researcher | Ericsson |

### 7.2.3. Additional Inputs

| Member | Title | Company |
|---|---|---|
| Lester Thomas | Chief IT Systems Architect | Vodafone Group |
| Ankur Goyal | Lead Consultant | Infosys |

| **Member** | **Title** | **Company** |
|---|---|---|
| Emmanuel A. Otchere | Chief Technical ExpertVP, Standards & Industry Development | Huawei |
| Min He | Chief Architect | Futurewei |