# TM Forum Introductory Guide

# Intent in Autonomous Networks

**IG1253**

| | |
|---|---|
| Maturity Level: General availability (GA) | Team Approved Date: 01-Aug-2022 |
| Release Status: Production | Approval Status: TM Forum Approved |
| Version 1.3.0 | 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

# List of Figures

# List of Tables

# Executive Summary

This document contains the proposal of the TM Forum Autonomous Network project on intent-driven operation.

The role of intent in autonomous networks is to communicate requirements, goals, constraints, and preferences to an autonomous system. This knowledge enables the system to evaluate the state of the controlled infrastructure and the utility of actions. It enables a level of autonomy where the system can adapt its behavior and generate new solutions instead of just following human-defined recipes and policies.

We introduce the Intent Management Function as an architectural building block for Intent-driven operations. Within an autonomous network, there will be multiple instances of intent management functions with different responsibilities. Intent management functions receive intents, make decisions about appropriate actions to improve intent fulfillment, control the execution of those actions, and report on the intent progress.

Intents are knowledge objects with a lifecycle that is actively managed by intent management functions. An intent management function in the intent owner role creates intents to define and communicate requirements to other subsystems and autonomous domains. An Intent Management Function in the role of intent handler receives intents and operates the domain that it is responsible for accordingly.

Intent Owners and Intent Handlers participate in intent life cycle management through the intent interface. It is introduced as an intent handling management service produced by the intent handler and consumed by the intent owner. The intent interface deals exclusively with Intent object lifecycle management tasks of intent objects. It does not contain any direct use cases or domain-specific aspects. However, these domain-specific aspects are described in the intent objects.

This document proposes the modeling of intent objects as knowledge graphs. An intent common model specifies domain-independent generic modeling artifacts such as the intent class and the expectation class. Intent objects contain a set of expectations that represent different and diverse types of requirements that allow all relevant concerns to be addressed.

This intent modeling proposal recommends the use of intents from a federation of models. While the intent common model contains general and domain-independent aspects, any number of intent extension and intent information models can be used. They are specific to a domain of intent handling and therefore define what the intent handler of that domain must understand and what the intent objects addressing that domain can express.

The proposed model federation is an invitation to other working groups and standards developing organizations (SDOs) to collaborate and work towards a consistent set of intent standards. The idea is that intent extension and information models can be defined by any organization and working group independent of the TM Forum. They would define the additional expressiveness needed in their chosen scope and domain, creating what are termed 'vocabularies'. By incorporating these intent extensions into the federation of models, the domain-specific intent objects remain compatible with each other so that common and domain-independent interfaces and lifecycles can be reused. We seek to avoid forcing incompatible intent standards into different and therefore costly implementations of management processes and interfaces.

This document is the main overview covering all proposed aspects of intent-driven operations. Detailed proposals and examples of key subtopics are provided through supplementary documents. Intent modeling is covered by IG253A, and intent lifecycle and interface are covered by IG1253C. Other supplemental documents will appear in future releases.

This document also introduces the ontology for intent management, known formally as the TM Forum Intent Ontology (TIO). It defines and categorizes intent management and provides a corresponding vocabulary for further modeling tasks, such as the expression of intent, intent reports, or intent manager capability profiles.

# Introduction

This document proposes a technical framework for realizing intent-driven operations within an autonomous network. The document covers the following main topics:

1. Definition of intent,
2. Discussion of the paradigm of intent based-operation,
3. Introduction of a generic architecture and principles of intent based operation,
4. How to model and express intent,
5. Discussion of modeling alternatives,
6. Life cycle management of intent objects,
7. The interface to communicate and manage intent,
8. Intent management capability management
9. Control loops with intent and intent reports,

# 1. Document overview

## 1.1. Scope and purpose

The purpose of IG1253 is to document and define intent-driven operations according to the architectural framework of the Autonomous Networks project. This includes a definition of intent, as well as the role of intent in autonomous operation and the operating principles associated with it. In addition, this set of standards defines the interface and API for communicating intent, the lifecycle management of intent objects, and the modeling principles of intent.

## 1.2. Overview

The use of Intent in autonomous networks is described and specified by a set of documents in which each individual document defines a separate aspect of Intent-driven operation:

IG1253 - Intent in Autonomous Networks

This is the main overview document. It contains a description of general definitions and operation principles. The sub-documents, IG1253A through to IG1253E, contain deeper views into some key topics.

TR290 - Intent Common Model

This document defines the modeling of intent and intent report objects as ontology graphs in RDF/RDFS. This document specifically defines the intent common model, which is providing the generic base vocabulary of intent expression.
TR290 has replaced IG1253A as reference for the intent common model.

TR291, TR921x - Intent Extension Models

This document defines intent extension models. These models are used within a model federation for intent expression. Intent extension models provide the vocabulary and define the respective semantics for domain specific, advanced and optional expression and modeling features. The introductory document TR291 establishes rules for intent extension models and provides an overview of proposed distinct models in the TR291 series. Sub documents have the ID TR291x, where x is a placeholder for letters A,B,C, etc. and define one distinct intent extension model each.

TR291 and its sub-documents replace IG1253B.

TR292 - Intent Management Ontology

The intent management ontology defines a model of general concepts of intent based operation and interaction of intent management functions over the intent interface. This establishes basic vocabulary used by the intent common model and other related models about intent.

TR292 combines the intent management ontology formerly defined in previous versions of IG1253 and the intent interface ontology formerly defined in IG1253C.

IG1253C - Intent life cycle management and Interface

This document describes the life cycle of intent and the interface used to perform life-cycle management. This includes procedures for communicating, modifying, and removing intent objects, as well as negotiating intent content and assess its feasibility. This document provides a high level introduction to the operation of the intent API that is planned to be published as TMF921.

IG1253D - Intent Manager Responsibility Scope and Capability

This document defines a registration and discovery mechanism about capabilities and scope of responsibility associated with an intent manager.

IG1253E - Use cases and examples (future release)

This document will contain a collection of use cases with detailed examples that demonstrate how to apply the principles and use the models and interfaces defined in the IG1253 set of documents.

# 2. Motivation for Intent

In recent years intent has become a widely discussed topic that is now considered to be essential for introducing automation and zero-touch autonomous operation. But why is intent given such a prominent place? What is its role and what does it add to the infrastructure and interfaces that was missing before? In order to analyze this, it helps to look back into the processes and tasks of manually operated infrastructure. IG1252 defines levels of autonomy and it proposes a methodology for assessing a systems' capability with respect to autonomous operation. It shows that the use of intent as base for autonomous operation correlates with an increase in autonomy level.

## 2.1. Fully manual operation:

Manual operation refers to a team of human technicians, who have the task of operating system, networks or infrastructure. They are involved in two essential tasks: intelligent decision and execution of actions. Intelligent decision refers to the process of collecting relevant information, analyzing the situation and planning suitable operational actions expected to improve the state of the operated system or network. Technicians, who are doing this, typically have a broad range of knowledge and information available to base their decisions on. They have access to extensive technical information such as the state of the systems and networks they operate as expressed by measured KPI and metrics. Furthermore, they also know what customers have ordered and therefore what level of service needs to be delivered to users.

The operations team of technicians has also quite a lot of contextual information that potentially impacts their decision. For example, they know the strategy and goals of the company they work for and the related business policies. This allows them, for example, to prioritize based on customer importance. They also know the budget situation of their unit which translates in allowed resource usage. Also, the goals and targets of their unit and priorities set by management will influence their decisions substantially. This will then result in prioritizing one action over another one as it is expected to better meet the business strategy and its related goals and targets.

Execution would be the implementation of these tasks in the infrastructure. Typically, this refers to tasks such as changing configuration data, replacing hardware or upgrading the network. These actions can mean an immediate reaction to solve an acute issue. They might also constitute a consolidated action plan targeting mid and long term improvements.

## 2.2. Operation with automated execution

The first approach towards autonomous operation is the introduction of automation. This refers to a system that is able to automatically execute a process without human involvement in every step and ideally without human involvement at all. Policies have a key role in realizing this level of automation. A policy is in this respect a rule or decision tree. It is initiated when a defined pre-condition or event applies, and it delivers an action or action plan that is then executed. The input to a policy-based decision is typically the technical the state of the system including the specifications of what customers have ordered. In general, all data and information systems are available to be used in policies, including inventories, analytics insights and measured KPI.

It is important to understand that a policy determining an action plan after being initiated from observation is not equivalent to the intelligent decision in manual operation. It is rather the automated execution of a pre-determined recipe. The developers, who have written the policies made all decisions at design time. They analyze the situations the system might be in and what a suitable action plan for each situation would be.

The policy developers need to define the conditions for invoking a policy. These are based on observable indicators, such as measured KPI or analytics insights. The invocation conditions allow the automated system to identify the situation and context the policy was made for. Furthermore, policy developers define the policy as an automated analysis and decision process. Typically, this is expressed through executable workflows including decision trees or sets of rules. When writing a policy, the policy developers has the same information and knowledge available like the technician in manual operation.

In operation, the automated system would invoke the execution of the right policy according to the situation conditions. This arrives eventually at an action to be executed. Policy-driven systems therefore automate the execution at run-time while the intelligent decision is still mostly human driven. The point of decision was moved to design time, while in life operation the automated system would apply the policies as behavior recipes.

This level of automation has already clear advantages with respect to lead time of fulfillment after a service order and 24/7 attention of the system with immediate reaction. This has already the potential to significantly increase the service quality and decrease the cost of operation. On the other hand, this automation also has limitations. The most significant limitation comes from the capability to adapt to changes and situations that were not explicitly considered at design time. In these situations, the automated system does not know what to do and must escalate for human support.

New situations can come from two direction. First, external environmental factors such as changing user behavior can constitute new system states. But also factors the operator directly controls can become new situations for the underlying automated infrastructure. Typical examples are the introduction of new products or new customizations of existing ones. Whenever a new situation occurs that was not explicitly foreseen at design time of the policies that control the automated processes, the automated system would fallback to at least partial human support. Keeping the system capable of staying fully automated in the new situation, would imply an adaptation of policies. The policy developer would introduce a future automated response to this new and similar situation.

In future networks, we expect diverse and customized service offerings based on technologies such as virtualization and slicing. In such environments a manual process for making all decisions and adaptations might not be capable enough to meet business demands. Especially lead-time in fulfillment as well as assurance might require further automation targeting automated adaptation.

Machine learning is often positioned as a technique to automate the way a technical system adapts its behavior. A typical machine-learned model is either used to implement analytics and therefore deliver insights to be consumed by policy logic, or it implements the policy by directly making decisions and determining suitable actions. This means the model can be a direct replacement of a manually designed policy. In any case, these types of machine-learned models are the result of statistics driven optimization processes and therefore follow evidence from data and observation. However, using machine learned policies in automated processes does not necessarily lead to a higher degree of autonomy.

Machine learning can be an excellent way to implement policies based on evidence seen in data, but the outcome of machine learning is still fully determined by human input and human intelligent decisions. Humans define cost and utility that makes the models converge as wanted. Humans also label training samples and therefore explicitly tell the wanted outcome the model is supposed to learn. In this respect, observed human behavior can also be direct input and then the preference is implied. In any case, it is still human intelligence that ultimately determines what decisions and actions machine learning-based policies produce.

## 2.3. Adaptive automation towards autonomy

A key goal of autonomous networks is to lessen the need for human involvement. Policy-driven operation as discussed in Chapter 2.2 allows leaving most standard and repetitive execution tasks to the automated system. Human supervision is always required to capture those exceptional cases where the automated system requires their direct involvement. Other than that, the human workforce would be able to shift their focus on tasks that still require their intelligence. They can, for example, do business planning, develop strategies, engage with customers, and define products and product customizations. The automated operation needs to follow these human decisions. This means that all artifacts that enable and control the automation must be kept up to date to ensure that the automation follows the revised strategies and is able to operate the new products. This might involve the modification of workflows, adaptation of policies and potentially re-training ML models with new data or revised utility definitions.

Utility is an important concept for autonomous systems. Utility refers to knowledge about what makes an outcome or situation preferential. This refers to preference in a global context. An action might be preferable because it helps to reach a goal, but a utility definition addresses the question if the action and therefore also the goal are addressing the right overall concerns and needs. Understanding utility allows the system to judge potential results and all goals defined and actions proposed to reach it. It enables the system to adapt its internal instrumental goals if needed. It allows to not only determine preferential actions that fulfill a goal, but it allows to reason about which goals are the most advantageous to pursue.

Transitioning the system from automation towards increasing degrees of autonomy would mean that more tasks become automated. This includes the tasks needed to keep the automated system aligned with business utility and strategies. The system would gradually take over the intelligent decision-making and therefore determine and adapt its operational solutions on its own and without human involvement. It would not just follow human-designed recipes but make new recipes itself if needed. A system like this would be able to determine if an action or plan is preferential or not and if it would improve or degrade the operational state with respect to a broad range of concerns. A system like this can then evaluate if a recipe for action is suitable even if the situation is new. It would have the knowledge to determine what actions would be preferential in the new situation. This requires that system has access to all relevant goals, requirements, and targets as well as constraints. The system would use knowledge about utility in its automated processes and adapt its behavior accordingly. Goals, requirements, constraints, and utility need to be presented to the system in a way that an automated logic can use them to arrive at potentially different, but still suitable decisions and actions. These are automated reasoning processes that adapt the system behavior automatically and to follow human determined and continuously changing requirements, goals and constraints.

## 2.4. The purpose of intent

The discussion in the previous sections has identified an essential enabler needed for autonomous operation: the automated system must *know* its requirements, goals, and constraints. The system can only adapt and follow business needs if it knows them. The autonomous system needs to know what it is expected to achieve by the service provider, who employs the autonomous system and the service provider's customers, who are served by it. This includes knowledge about expectations including hard requirements, but also about preferences and priorities.

Knowledge about these topics can change dynamically because it ultimately originates from dynamic concerns such as service provider strategy and customer need. The autonomous system always needs to be kept up to date about their expectations. Only by knowing the expectations the autonomous system has a chance to meet them.

Furthermore, this knowledge must be presented to the system in a way that enables automated reasoning processes to translate them into adapted system behavior. This means, knowledge about the expectations towards the system need to be formally expressed, communicated, and managed.

The purpose of intent is to define and communicate knowledge about expectations to a system in a way that allows automated processes to reason about it and derive suitable decisions and actions.

Intent is, in this respect, the knowledge element for communicating expectation. It allows the autonomous system to know requirements, goals and constraints that are the foundation for all action. It is the reason for a policy to prefer one possible action over other options. Intent is the foundation for prioritizing decisions and optimization actions. Intent determines a customer's needs and the service provider's contractual obligations. It enables exploration of potential solution options and evaluation of actuation strategies in order to find one that delivers the best available business result. Intent-driven operation refers to models, interfaces and architectures for managing this knowledge and to operate a system accordingly.

The knowledge of intent enables automation of intelligent decisions that were still entirely human driven in systems with automated execution. It allows evaluating situations the system is in and prioritize actions that transition the system's operation strategy into a preferable direction. Intent allows to communicate what is preferable and what needs to be avoided. It allows the receiving system to understand what it is expected to achieve. It introduces a notion of utility. It allows the source of intent to communicate its utility model. This enables intelligent judgment about the situations the autonomous system observes and actions it plans to do. The system is effectively able to determine the utility of its solution options. It can judge if its polices produce preferential outcomes and modify them if not. An intent-driven system is therefore able to not just blindly follow human-determined solution recipes. It can modify them and make its own.

While using intent is a necessary prerequisite for implementing advanced levels of autonomous operation through self-adaptation capabilities, intent is also already useful in automated execution and policy-driven operation. The use of intent does not preclude the use of policy based systems. Here, intent can be used to determine policy triggers and its detailed expectations can be consumed in decision trees that lead to actions. This means a developer can diversify policies by considering changing requirements and goals.

# 3. Definition of Intent

## 3.1. History of intent definition

Intent was first introduced around 2015 in the context of SDN controllers. At that time IETF defined intent as

"*… an abstract, high-level policy used to operate the network*" [rfc7575].

This definition implies a strong relationship between intent and policy. The guiding idea is that intent directly translates into choosing the right policy that then operates the system accordingly. The chosen policy is, in this respect, a pre-defined recipe for the set of requirements chosen by the given intent. This automation strategy means that every selectable variant of requirements must be matched with a policy available in the system. Intent is therefore an expression of chosen requirements that are tightly coupled with a matching policy for controlling its fulfillment within the automated operational processes and therefore its implementation with the controlled resources.

In the meantime, the understanding of intent has evolved, and this is reflected in a more recent definition of intent by IETF from 2020. Intent is now defined as

"*… a set of operational goals that a network should meet and outcomes that a network is supposed to deliver, defined in a declarative manner without specifying how to achieve or implement them*" [ibn].

As policies determine the actions taken by the system, they are considered part of the implementation. By excluding explicitly any specification of how to achieve or implement an operational goal, the newer definition of intent explicitly excludes mandating policies. It also excludes mandating hard-coded logic or artifacts such as rules and workflows that define decision trees and decision-making processes. Intent is therefore purely the specification of requirements and goals separated from all implementation artifacts.

In the Autonomous Networks Project at TM Forum, we propose to move forward with a conceptual understanding of intent following the definition from IETF in 2020 [ibn]. This proposal enforces a strict separation between intents being purely an expression of requirements and other implementation artifacts of any kind.

## 3.2. Definition of intent

The definition of intent proposed by the Autonomous Networks Project follows the analysis and findings presented in Chapters 2 and 3.1:

**"Intent is the formal specification of all expectations including requirements, goals, and constraints given to a technical system"**

This definition is inspired by and compatible with the definition IETF has published in 2020 [ibn]. The definition associate's intent with goals, requirements and constraints provided in a declarative way. Intent constitutes and expresses knowledge about these concerns and enables sharing this knowledge between the originator and receiver of the intent.

Furthermore, this definition excludes all imperative implementation and solution aspects form the intent itself. Intent is therefore purely an expression of what needs to be achieved or avoided or what outcome is more or less preferred, rather than indicating how and by which strategies and actions this can be realized. In this respect, artifacts such as policies, workflows, rules, decision trees and other ways to express and implement a solution strategy, make decisions and execute actions are still very much-needed to realize intent-driven autonomous systems. They are however strictly separated from the intent expression. This understanding of intent implies implementation of Intent-driven operation that strengthens important system design concepts such as strong separation of concerns between sub-systems with full encapsulation of solution implementations.

The notion that intent is a specification of expectation reflects the viewpoint of humans as external supervisors of the autonomous system. They expect the system to fulfill their needs. The system has to meet their expectations and intent is the expression of their needs. Intent can in this respect originate directly from humans. These humans are, for example, customers or operator personnel using intent to directly communicate with the autonomous system through intent, and they expect the system to meet these intents utilizing the underlying infrastructure and its resources in a suitable way.

Furthermore, intent is also generated internally within the autonomous system. It is used between the sub-systems and system layers to influence details of their specific wanted behavior and thus contribute to the overall fulfillment of human expectation. Intent coming from external sources into the autonomous system constitute the terminal goals of the system as a whole. The autonomous system would then derive instrumental goals reflecting a solution strategy and the detailed needs and concerns involved implementing this strategy and ultimately achieving its terminal goals. Internally used intent is an expression of these instrumental goals and a mechanism to distribute them to the responsible sub-systems. The goal breakdown is typically applied in multiple steps transforming global terminal goals into its localized and detailed consequences. Intent is the way to express these goals on every detail level and with every granularity needed. This also means that internal intents are always used in the context of and can be traced back to those intent, which directly express human needs. Therefore, also internal intents express expectation. The communication and management of intent is a central topic of IG1253. It is covering multiple aspects such as a generic system architecture of intent-driven operation, life-cycle of intent as knowledge objects and the establishment of control loops steered by intent. IG1253C introduces the interface to communicate and manage intent as part of lifecycle management processes.

While intent might originate from human input or being created by automated processes within the autonomous system, it is typically received by a technical system. Intent is usually not meant to be consumed or acted upon by a human receiver. Nevertheless, Intent might also be used by the system to delegate tasks to humans for execution. The details of using intent this way or if alternatives, such as the use of a dedicated domain specific language would be preferential for these use cases will be explored in future work. Also, in cases when the autonomous system breaks and a human team need to take over, humans would need to act on the intents. But this would be considered an exceptional situation rather than usual operation.

Humans might have formulated intent, and they might monitor the automated operation including intents being used between sub-systems. In this respect it is preferential that the methods and techniques chosen to express intent are reasonably intuitive for a human to read and understand. On the other hand, it is a system receiving the intent and automatically operating based in it. This is a challenging task, and it is therefore highly important that the chosen techniques and models for intent expression facilitate

a practical implementation of a technical system with automated processes able to translate intent into solution strategies and operational actions.

This points at one of the most important aspects of intent definition: the use of formally defined models. This means intent would be expressed with formally defined and complete semantics and vocabulary. There must not be ambiguity in the meaning of intent. The sender and receiver of intent must be in perfect agreement about its interpretation. And it must be possible to derive this agreement entirely from formally specified and complete semantics in the underlying common and information models. If interpretation of an intent cannot be derived from the formal modeling, the consequence would be that human consultation is needed to clarify ambiguities. This would violate autonomy. In the worst case ambiguity might lead to diverging interpretations of the same intent by multiple involved systems causing unwanted and incorrect system behavior. It is therefore essential that intent follows formal models that comprehensively and unambiguously cover all needed expressiveness. The proposed intent modeling approach is described in Chapter 13 and specified in detail in IG1253A.

In this respect, intent-driven on natural language and other domain specific languages is not in scope of IG1253. Especially natural language is inherently ambiguous and needs interpretation considering context and shared assumptions. We acknowledge however that there are use cases and good reasons for formulating intent by using natural and domain specific languages. Their usage in relation to intent operation and modeling defined in IG1253 is discussed in Chapter 15.

# 4. Properties of intent

## 4.1. Declarative goals and utility: the wanted state

Following the definition of intent presented in Chapter 3.2, an intent object is a collection of distinct expectations. They express a variety of requirements, goals and constraints. An individual intent may contain a variety of different expectations. For example, the expectation to deliver a service to users would contain functional and non-functional aspects of the service itself and the targeted usage scenario. It can include minimum performance levels as well as usage limitations such as geographical availability. All these aspects are considered in intent modeling to be distinct expectations.

Intent is solely declarative in the sense that it only specifies wanted outcomes versus outcomes that need to be avoided. This can include quantitative specifications. For example, a goal can be set by defining target values or value ranges using KPI and metrics. Depending on the targeted scope and subject of the intent, the definition can be high-level and abstract, or it can be technical and detailed. For example, a business level intent can specify the need to make a financial gain from autonomously managed SLAs. In this context a detailed target can be set, such as a required margin of 10%. An example of a lower level technical intent would be to guarantee a minimum latency on a particular network link.

Both examples specify a wanted outcome or state without specifying how to reach it. It is entirely up to the intent receiving system to find a strategy and plan actions that would achieve what the intent is asking for. This implies a level of encapsulation in which the system implementing the needed operation processes does not expose these processes through its interface. This means an intent-driven order would only specify what is needed and not also trigger the process that implements how this is achieved. It is purely a decision of the system that receives the intent to decide how to act and which processes need to be invoked to fulfill the requirements. This is a significant difference between intent-driven operation and other typical interfaces in telecommunication. Telecommunication interfaces often explicitly invoke processes, which are exposed and implemented by distinct functions and services.

Intent does not imply the start of a process, because this is considered imperative prescription of what to do. All imperative specifications are explicitly excluded from being part of an intent object. Imperative specifications would include actions that need to be taken or avoided. Also, implying to invoke specific processes or workflows are imperative specifications. This includes mechanism such as policy triggers. All these specifications are not part of an intent expression. They are conceptually excluded and consequently not covered by intent modeling. Intent rather only communicates knowledge about requirements and leaves the decision what processes to invoke to the intent receiving system.

As intent leaves the actions to be taken entirely open, the receiving autonomous operation system has in principle a great amount of freedom to apply already known strategies to develop and explore new solutions. Enabling this is a key property of the intent-driven operations mechanism. It is therefore ideal if intent expresses utility. Utility refers to knowledge about what properties of an outcome are preferential and which are not. This is not just setting a goal that the autonomous system shall reach. It allows evaluating if reaching these particular goals is actually the best solution of if another goal would lead to a better overall result. Utility knowledge therefore enables self-reflective operation with the capability to adapt system behavior autonomously.

## 4.2. Composible and additive

An autonomous system that operates a complex and shared domain would need to fulfill multiple requirements. This includes to reach a potentially large number of goals, obey a complex set of constraints and consider preferences. This constitutes a pool of requirements the autonomous system is operating against.

Intents are knowledge objects that define a set of these requirements. Setting an intent therefore means to add to the pool of requirements. Deleting an intent means to remove requirements from the pool. The intent mechanism therefore manages the global requirements of an autonomous system as changing its requirements implicitly alters the system behavior.

Intent and the additional requirements expressed by it can originate from many sources. This means that they might overlap or even contradict. This cannot be avoided on the level of intent, because even contradicting requirements and goals are still valid and express an important concern of another system or a human user. This means the autonomous system operating based on intent would need to be able to prioritize based on utility. It might also not be able to fulfill all its requirements at once in all situations.

## 4.3. Persistent and lifecycle managed

Intents are knowledge objects that communicate requirements, goals and constraints as well as preferences. This means using intent establishes a requirement the receiving system has to fulfill. This requirement stays valid until it is removed by deleting the intent. Intent objects therefore have a lifecycle that is actively managed. The intent interface defines the procedures for executing this lifecycle management. The intent lifecycle is explained in detail in Chapter 8, and it is the base of the intent interface defined in IG1253C.

This also means that the use of intent includes assurance aspects. Intent defines not only what to deliver, but also what to assure. In this respect, an intent is not done, for example when it is first fulfilled by meeting its requirements. Intent implies that the requirements stay fulfilled until the intent is removed.

## 4.4. Infrastructure agnostic and portable

Intent as defined in this document is a key mechanism used in the communication between the layers in the autonomous networks framework (ANF). Furthermore, it is used in the interaction between autonomous domains. This means, intent will need to cross the borders of major sub-systems and platforms, where often solutions from different system vendors are used and required to interact flawlessly. This need is addressed by defining a common interface for intent lifecycle management as well as a common modeling approach of intent objects. Modeling of intent is described in Chapter 13 and specified in detail in IG1253A and a future IG1253B once it is released.

## 4.5. Measurable and grounded in data

An intent is only useful if all aspects it specifies can be observed. A system can only control what it is able to measure. This means that a system that operated based on intent would require to be connected to data and knowledge sources. This includes, for example, metrics and KPIs being measured, aggregated and calculated or analytics functions providing insights to the system.

As intent can change dynamically, the receiving system might need to adapt and measure whatever the intent is expressing. This is mainly imposing a limitation on the allowed range of intent expression. Intent can only be handled successfully if the receiving system has the means to observe whatever the intent is stating. The implementation of autonomous systems might differ considerably with respect to the range of intent expressions it can operate based on its capability to measure and observe. This is addressed through intent handler capability management as discussed in Chapter 12.

# 5. Expressiveness of intent

Every concern relevant to an operator or their customers or other involved parties, such as government regulators, can directly or indirectly constitute intent. From the point of view of the autonomous system, intent is all it needs to know about the concerns of relevant external parties so that it operate and serve them as expected or as close to their expectations as possible. This chapter discusses use cases for using intent. This includes the concerns of involved parties and what knowledge intent would need to express in order to provide sufficient input to the autonomous network. It also includes the needed expressiveness for intent used internally within the autonomous network to coordinate the contributions of its sub-systems in meeting the original expectations.

This chapter provides examples for needed intents mainly in the context of telecommunication services, network operation and business concerns of network operators and their customers. In this chapter we mainly present typical examples from this range of use cases. They are used as inspiration and guidance for technical proposals of IG1253, but they do not constitute an exclusive or complete list of use cases and concerns intent is used for. Extension and use in further domains is explicitly encouraged and also considered in the proposals and the resulting technical solution.

We expect that higher degrees of autonomy and the implied reduced involvement of humans in the operation will cause a gradual shift of concerns from human to machine operation. This will consequently imply that further expressiveness is required by intent to communicate the specific goals and requirements related to that concern. Also, extension of autonomous operation into new domains will lead to further concerns to be addressed by intent.

## 5.1. SLA negotiations and agreement

Network operators are in the business of selling services to their customers. This typically involves a contract in the form of an SLA stating all agreed functional and nonfunctional properties of the service as well as the terms of compensation including penalties in case of contract breach. An autonomous system for customer engagement might include partly or fully automated negotiation and acceptance of contracts.

A possible scenario would be that both, the operator and its customer have fully automated systems for SLA contracting. These systems automatically create offers, determine if the offer would fulfill the needs and negotiate compensation. Humans are not directly involved in the process of contract negotiation, but they can use intent to steer the process by setting goals and constraints. In an alternative scenario only the operator's system is automated, and it communicated with the customer's personnel through self-service frontend.

The operators would use intent to steer the contracting on their side of the negotiation. These are intent given to the autonomous contract management and customer engagement system to steer its behavior. The operator's concern expressed by intent could, for example, be to make a financial gain. The intent used for this can formulate a goal for automated contracting about the expectation that it generate a financial margin with the contracts it accepts. A financial margin across all contracts would be the measurable metric used for setting this goal. This intent would then be input for generating service offers and in the logic for accepting proposals by the customer.

## 5.2. Delivery of user services

After the operator and its customer have agreed on the details of a service, an autonomous network would utilize the resources of the operator's infrastructure to deliver these services. In order to do this, the autonomous network needs to know what shall be delivered to which group of users and if there are special limitations and concerns to consider. Intent would for example be used between the operator's contract management system and the autonomous operation system. This is the interface between business operation and service operation in the Autonomous Networks Framework. Intent would typically need to express the following:

**Functional requirements:**
Specify, for example, which services need to be delivered and what function do they need to provide to the user.

**Non-functional requirements:**
These are typically targeted regarding performance, availability and user experience. KPI and metrics would be used to express them.

**Constraints and inter-dependencies:**
Are there any special concerns or requirements that a solution would need to obey. For example, privacy and security concerns of the customer might need to be addressed considering through multi tenancy and security levels. Another example would be that for legal reasons all data and service instances need to stay in a particular geographical location.

## 5.3. Behavior of resource services

An essential task of operational process within a service provider would be to break down higher level requirements expressing business needs into deployments and configurations of technical assets such as resources and services. For example, software instances need to be deployed following a target topology, network functions need to be connected to each other with sufficiently configured channels and the entire setup need to be assured involving measurement of KPI, analytics and detection of issues.

Intent can be used to steer this level of operation with increasing levels of technical details. These intents are on a lower level in the operation software stack. But they are very similar to the intents used to describe requirements for user services. They would express what needs to be delivered. This would be an expectation towards a sub-system, rather than the service provider and its autonomous network as a whole. These intents can also express non-functional requirements and constraints and inter-dependencies. The difference is that artifacts and metrics used in lower level operation are not directly accessible to the customer or its users. They represent the means by which the autonomous network satisfies customer needs, but these details are not exposed to the layers above and the customer. In other words, customers would only see the services they have ordered and the properties they have directly agreed to, but not the lower layer details of how their services are realized.

This indicates that intent is typically used in a hierarchical way where the intent of user services is broken down into intent about resource services and the behavior of the underlying infrastructure. The autonomous networks framework models this by distinguishing business intent, service intent and resource intent. Please note that there can be more than three layers of intent being used in a practical autonomous system.

A user might have ordered a communication service with agreed throughput and latency. These are the KPI about the direct user experience and these are stated in the intent about the service. In the autonomous operation this service might be realized by selecting and deploying network functions in a datacenter and by setting up network slices to interconnect the network functions. An intent about resource services would describe the deployment requirements of the network function instances as well as the required latency and throughput of the involved network slices. These are the KPI targets needed from the involved resources the system needs to deliver the experience KPIs agreed with the user.

## 5.4. Regulatory and legislative requirements

It is the mission of regulators to set and enforce rules in the market they oversee. Service providers must follow these rules to avoid penalties and keep licenses. Intent can be the mechanism to directly distribute and communicate the rules from the regulator to all service providers. In this scenario the regulator actively participates in intent-driven operation. The respective reports on intent fulfillment from the service provider to the regulator can then be interpreted as compliance statements. The regulator becomes an intent using third party next to the operator and the customer. As regulatory requirements are legally binding and globally applicable they imply utility with a broad scope and high importance.

The direct participation of the regulator in intent-driven operation introduces the ability to react quickly and change market rules dynamically. A possible use case would be exceptional cases of reaction to disasters in which the service providers might need to temporarily change service prioritization. Using automatically distributed intent considerably reduces the reaction lead time.

In another scenario the regulator is not directly encoding intent but, intent can still be used to configure and steer the autonomous operation within a service provider domain. It would be the service provider itself creating the intent-driven on the regulatory rules. As the service provider is accountable to meet the rules it is therefore its responsibility to follow up on changes and modify intent accordingly. This is typically a manual process that introduces delays. But usually rule changes are announced in advance with a clear deadline for implementing new rules. So, the service provider would have sufficient time to review and adapt.

Nevertheless, intent being used would allow the autonomous network to directly access regulatory rules. The rules would be formulated in a way a machine can read and reason about them. Thus, the autonomous network can consider regulatory aspects in all its solution decisions and actuation. Regulatory rules become yet another set of requirements to follow. Like other intent regulatory rules would therefore be considered within all autonomous operational processes. This mechanism would keep the autonomous network compliant to legal requirements and regulatory rules.

## 5.5. Solution Bias

Prioritization and optimization within operations processes follows a bias introduced through requirements and goals as well as the available actions and solution strategies. This means, what a system prioritizes and chooses to do and the outcomes it produces is ideally only going into a wanted direction. However, this cannot be guaranteed. Models and policies might contain errors or reflect the opinion of the human developer and data scientist, who has produced them. And this opinion might

not be a full match with the service providers goals and values. In machine learning there might also be bias in training samples, which then constitute bias in the resulting models.

Depending on the market and business environment of the service provider, certain bias might not be significant, but it can also bear a huge risk. For example, if a service provider is exhibiting social, racial or religious bias in the way their autonomous network decides and behaves, this service provider might not be perceived favorably with significant impact on business results. Detecting unwanted bias in the behavior of the autonomous network and eliminating this bias is therefore a critical business capability of a service provider. This typically requires putting policies and models in place, which are able to detect bias in operation including techniques to measure it.

Intent can play the role of steering operational decision-making by introducing requirements on bias. Techniques and metrics introduced to measure bias are the tools available to formulate a respective goal expression. Using this intent as additional requirement in intent handling would make bias awareness part of all operational processes. It would imply that all decisions and actions and their respective consequences would be checked against all intent and thus also the bias avoidance requirements.

This mechanism also allows introduction of special attention to types of bias with raised sensitivity in the local market.

## 5.6. Limit Risk Taking

Every operational action being done or not bears a certain risk to disrupt operation and failure to deliver agreed services. In manual operation the assessment of risk associated with an action is one of the most important tasks. It not only refers to the risk of action failing or not delivering the promised impact, but also to the consequences this single action might have on the network and operator business as a whole. Humans are usually good at considering a broad spectrum of consequences and therefore avoid too risky action strategies while finding a balance between risk versus potential gain and utility of the action.

An autonomous system needs to perform similar risk assessments and consider them in their decisions and actions. At least this would be recommended for systems with broader scopes and authority to perform nontrivial tasks with potentially huge impacts.

One way of limiting risky actions would be to explicitly limit the options available to the autonomous system through rules or policies. This requires however that it is possible to formulate comprehensive rules considering all situations. Explicit limits might also inhibit potentially very preferential actions. Especially in systems with higher degrees of AI, a tight frame of limitations counteracts the AI ability to find new solutions.

If a system is able to determine and manage risk, the service provider might still want to determine the allowed level of autonomous risk taking. Practically higher risk might be allowed if at the same time the potential gain is similarly huge. Thus, the system needs indication for how to do a trade-off between expected gain and imposed risk. Intent can be the mechanism the service provider is using to formulate respective requirements and constraints. For example, metrics for risk assessment can be used to formulate respective expectations.

## 5.7. Common sense

Humans have common sense and machines do not. This refers to a common understanding and agreement about what are right and wrong action and preferences. This common understanding means that requirements and best practices are followed, even if they are not explicitly communicated. For example, to a human it is common sense to operate a network and delivering services to customers while reducing the number of resources needed. Humans understand the overall need for the operator to make financial gain, they understand that resources are a cost factor and that therefore keeping resources usage optimized has a positive effect on the overall goals.

An autonomous system does not have common sense. It can only consider explicitly stated dependencies and goals. If a goal, such as saving of resources is important, and usually it is, then the system needs to be told explicitly to make the right choices in its solution strategies and actions. There are typically two ways to reach the right behavior. These requirements are built into the code or policies of the autonomous system. This way a developer has considered all relevant common sense and therefore also all resulting actions of the system consider it. This corresponds to operation with automated execution as discussed in Chapter 2.2.

Alternatively, the concerns that typically constitute common sense can be expressed as intent and become explicit goals. For example, intent can set a goal on resource saving. A good metric for expressing this goal would be needed to avoid misinterpretation. It is for example not a good idea to set a goal on delivering services with the least resources. Not delivering the service at all would then become an attractive option. A better metric would be to maximize resource utilization. This refers to resources being reserved and actually used. The rationale is that it is a good idea to use resources as long as they are productive and contribute to satisfying customer needs and generate income.

This discussion shows that common sense can be introduced using intent. It also shows that requirements and goals need to be introduced with full awareness of their interpretation by the system and if they imply also unwanted outcomes or make them look more preferential.

## 5.8. Communicate and escalate to humans

Service providers need to be kept informed about the performance of their autonomous operation. This would allow them to identify shortcomings and take actions. They might for example invest into the infrastructure to counteract frequent resource shortages, or they might order improvement of AI models and policies to be developed. Ultimately, the human personnel needs to know when the autonomous system is failing, and they need to step in and take over the operation at least partly.

Intent reports are a basic mechanism to achieve that. They allow establishing continuous reporting about each intent and its operational status and success. The exact conditions for reporting are in this respect additional expectations required within the intent itself. It allows specification of what and when to report. This reporting mechanism can be combined with frontends for intuitive presentation and therefore allow already continuous and detailed information.

Another way the system can interact with the human workforce would be for escalations. This refers to situations where the autonomous system detects that it needs human input to operate. This can for example be the case if the system is missing essential information and it wants to request clarification. Another example

would be that the system did not find a suitable action to mitigate a problem automatically. It would escalate this situation to the attention of a human technician. An escalation might also be chosen if all available actions are considered too risky and human approval is needed.

Setting conditions for escalating or for asking human approval can be subject to intent. The service provider can therefore use intent to steer the wanted interaction. Setting these conditions for example based on metrics for risk and gain assessment allows to precisely limit autonomous risk taking and only bring significant topics and situations to the attention of the human workforce.

This mechanism based on intent for setting the escalation and approval conditions would also allow the gradual assignment of more authority to fully automated operation. Starting with strict rules that bring a lot of decisions to human approval the system might prove itself by consistently proposing good solutions. Based on this the service provider can choose to offload more situations to fully autonomous operation.

## 5.9. Customer and resource value

Service providers usually differentiate their customers. Not all customers and not all users are equal. Some might have premium contracts promising better experience and priority, while others have chosen a more budget friendly option that comes with lower guaranteed levels of experience. This differentiation is important for the decisions and actions of an autonomous system.

Intent can be used to convey priority. This can be done implicitly by distinguishing services for the respective user groups and assigning different requirements and goals in the intents for the services. It is also possible to introduce explicit categorization of user service levels, for example by distinguishing silver, gold and platinum services or customers.

Typically, the categorization of customers and users follows the business value of their contracts. This would be handled primarily on business operation level, where all financial aspects of the service contracts and SLA are managed. Service and resource operation are not involved although they need to make decisions accordingly. Adding indications of relative customer value or service value into the intents would be an expression of utility. It can help in operational decisions and especially if prioritization is needed to optimally use limited resources.

Similarly, resources have a business value, and they should be assigned so that the income they generate exceeds the total costs of use and ownership. Also, here intent can help to convey business considerations and priorities to the decision-making in lower layers or the operations stack.

Customer value as well as resource value are examples of metrics that can be used in intents to provide hints about the optimal operational state.

## 5.10. Default or minimum requirements

The service provider might want to set minimum requirements for operation. These are meant to be globally applied to all services provided. These can be functional or non-functional requirements. For example, the service provider might choose a marketing strategy to distinguish itself through security. A concrete action to back up this claim would be that all network links are encrypted.

Intent can be used to introduce this requirement into the autonomous network operation. It is a global default. This means that it applies to all delivered services irrespective if their service specific intent requires it or not. This global requirement forces to autonomous network to only consider solutions that include encryption.

An example of a global non-functional intent would be the setting of a minimum availability requirement for services. This would force fulfilling service intent by choosing more reliable deployment options even if they are more expensive.

# 6. Categorization of intent

Categorization of intent helps humans to understand the meaning of intent content and the context in which the intent is used. It indicates who has created the intent and for what reason it is sent to a particular sub-system, system layer or autonomous domain. Intent types and categories are labels to summarize the intent use and content. A business operations type of intent is a business operations intent, because it uses business level terminology and metrics to formulate the expectation details and because it is given to an intent management function with a responsibility scope of business operations.

The system, which receives an intent, operates based on the detailed requirements and goals expressed within and through a set of expectations. Explicit intent typing is a summary of the same information. An intent type would therefore at best contain redundant but incomplete information. It does not imply any additional meaning that is not yet represented better and more detailed by the given expectations. Intent handler logic would be built solely based on the expectation expressions.

Also, distribution of intent to the right handlers by matching handler capability with wanted intent expressiveness cannot be based on intent types. A detailed match between the level supported expectation and information objects is required to control compatibility of the intent with handler capability.

For these reasons, the proposals of IG1253 do not introduce intent types. In particular, the models that specify intent expressiveness do explicitly avoid sub-classes of the intent class and properties for assigning types and categories to intent objects. It is however possible to mention types in comments within intent objects. However, comments are solely for human guidance and documentation and are not processed by intent handlers as part of their operational processes.

While intent types are not recommended being part of the formal modeling of intent objects, we recommend introducing them as a tool to describe the autonomous system. In this respect the autonomous networks framework distinguishes business intent, service intent and resource intent. This categorizes intent being used between and within the major architectural layers of an autonomous network. For example, a business intent is in this respect any intent that carries requirements, goals and constraints targeted at functions and tasks within the business layer.

There are multiple dimensions for categorizing intent. Intent can address certain concerns and therefore target a specific functional domain. Intent expressing requirements for automated contract negotiations in BSS are certainly different from intent imposing radio coverage requirements in RAN management. What distinguishes these examples are the information models used to express expectations. The information models are domain specific which gives the intent a natural scope and implies a type.

Depending on which aspect and concern to describe and discuss a different set of intent categories might be used. An intent can therefore have multiple types at once coming from different dimensions of categorization.

Here are a few examples of dimensions of intent categorization. The list can be extended as needed:

1. By targeted responsibility scope
   This can be the domain and layer in the autonomous network framework, or it can be the intent handling scope as introduced in Chapter 11.
2. By concerns addressed with intent. Multiples are possibly addressed by a single intent at once.
   Examples: Service delivery, resource behavior, regulatory compliance, ...
3. By origin type
   This categorization is derived from the category of the intent source. For example, the type of entity that created the intent: Human or another system.
4. By origin role
   The role of the entity the intent originates from: product manager, customer, user, technician, ...
5. ... the list can be extended as needed ...

Intent objects are a collection of distinct expectations. This means a single intent can potentially address many concerns at once and therefore partially meet the conditions of several intent categories.

The dimensions and categories can be extended if needed for explaining further aspects of the operation. There is already sophisticated work on intent types and categories done by other work groups and organization. Their proposal is often based on domain expertise. We propose to use these proposals of intent categorization as needed as long as intent typing is not introduced in the models for creating intent objects.

# 7. Principles of Intent-driven operation

## 7.1. Intent Management function

We introduce the intent management function as the entity that operates an autonomous system by using intent. It can assume the role of an intent owner or intent handler or both according to intent life-cycle management and interface as defined in Chapter 8 and IG1253C.

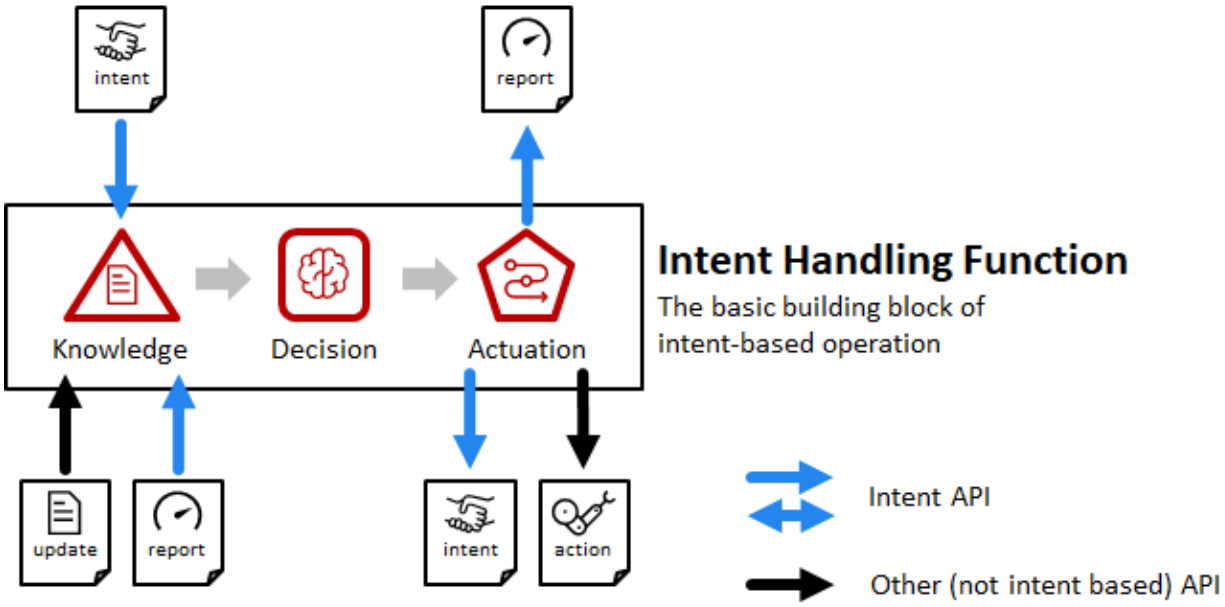


**Figure 7-1: Intent management function**

Figure 7-1 introduces the intent management function. Without implying a specific implementation, we assume that intent management functions operate based on knowledge, make decisions about actions to be taken and has the means to execute the chosen actions.

Knowledge refers to knowing the operational goals and requirements as specified by intent. The intent management function is an endpoint of the intent interface through which it received the intent it is supposed to handle, thus base its operational decision and actions on.

Knowledge also means knowing the state of the system or domain for which an instance of the intent management function has the responsibility to operate. While intent specifies the wanted state to be in, measurements and analytics results determine the current state. The decision of the intent management function is mainly about closing the gap between the current measured and wanted state.

The intent management function decides about suitable actions needed to fulfill the intent. The chosen action plan can involve the definition of further intent used to communicate requirements and goals to other sub-systems. This means an intent management function can act by defining intent. In this case this instance of the intent management becomes an intent owner. It can however also act through conventional interfaces for example by invoking processes or changing system configuration. The

actualization of the intent management function would then implement all needed interfaces. This means that an intent management function interacts with and relies on other functions of the domain it operates. This domain responsibility makes instances of the intent management function highly contextual. They have a defined and exclusive scope of responsibilities. The scoping of intent handling is further discussed in Chapter 11.

## 7.2. Intent reporting

Intent reports are exchanged between intent management functions for reporting on status and success of intent handling. Intent management function can have the role of intent handler or intent owner. Intent objects are created by the intent owner and sent to the intent handler. The intent handler operates based on this intent and reports back to the intent owner about progress and success. The roles of intent management functions are discussed in further detail in the context of intent life cycle management in Chapter 8.

Intent reports are therefore knowledge objects that always correspond to an individual intent object. If an intent is sent by an intent owner to an intent handler, the intent handler will start sending reports back to the owner. This means for each individual intent object there will be a sequence of reports directly related to this intent.

Intent reports are pushed by the reporting intent handler rather than being pulled by the intent owner. When and why to create reports is determined by conditions defined within the intent through reporting expectations. The intent handler has access to detailed state information and measurements of the domain or system it operates. This means only intent handlers can detect if an intent is violated. Therefore, only a push mechanism for reporting would allow immediate reporting of major events such as intent degradation. While intent owners define through intent objects what operational aspects are relevant, an intent handler would use its domain knowledge to report on these aspects and only on these aspects. Intent handlers are in this respect knowledge aggregator and relevance filter for the intent owners.

Intent report is the only mechanism provided by intent based operation to inform the intent owner about the system state and system compliance to the intent. This encapsulates the lower level details about resources and only communicate about the requirement details with the intent owner. This creates good separation of concerns and clear separation of authority and responsibility between domains. However, it is allowed that an intent manager implements additional interfaces directly with measurement, analytics and any information system or inventory available. This can be done in addition to intent based operation with its intent reporting mechanism, but it is outside the scope of specifications and standards about intent and its management.

Intent reports also play a role in the intent life-cycle management. They also communicate handling status and progress. Communication from the owner about setting, modification or removal of intent is answered with an intent report. In this respect the intent report also carries information about handler decisions such as acceptance or rejection of intent and the rejection reason.

The intent management function that has sent the intent is the receiver of the associated reports. Although intent reports are pushed by the intent handler, the receiver of the report has full control of the reporting. It specifies the reporting conditions through the intent. If additional or less reporting is needed, the intent owner can modify the intent and the reporting expectations within. The intent handler would adjust its reporting accordingly.

## 7.3. Intent in the autonomous network framework (ANF)

The intent management function is a generic architectural component for the realization of an autonomous network's framework. All intent is created, managed and operated by intent management functions and through the intent interface. Thus realizes an intent overlay across architectural layers and throughout autonomous domains.

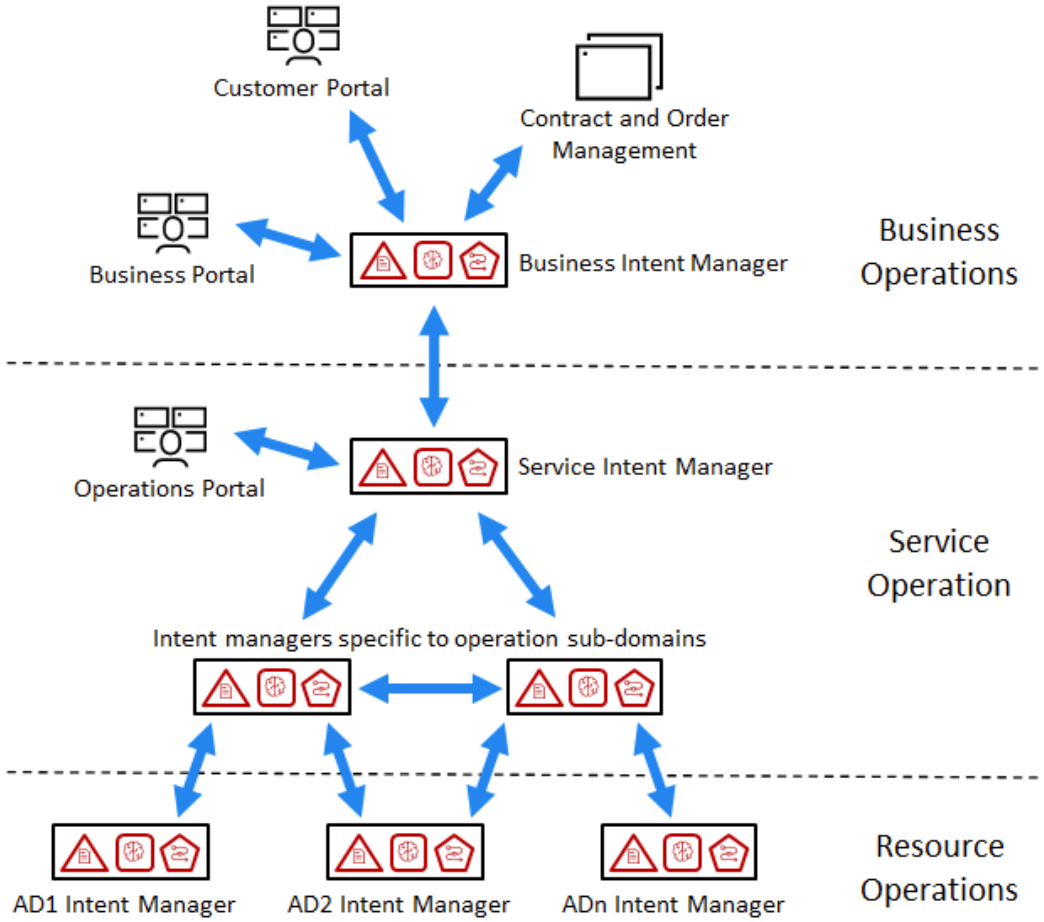


**Figure 7-2: Example Intent-driven operation within the autonomous network's framework**

Figure 7-2 shows an example of multiple intent management functions with their individual responsibility scopes. Note that the arrow representing the intent interface is bi-directional in this picture. This includes the direction of intent setting as well as the direction of intent reporting between intent management functions. All procedures proposed on this interface are discussed in detail in Chapter 10 and IG1253C.

In this example there is one intent handling function in business operation. It receives its intent from contract and order management. This intent might originate from SLAs, and it reflects the contractual obligation towards a customer and the needs of the respective users. While contract and order management might be an automated system that generates intent, further intent can come directly from human personnel of the customer. Also, the operator's personnel can inject further intent directly into the autonomous system. Human intent setting would be done through respective frontend. In this example we show customer and business portals. Both are human interface frontend allowing direct or indirect intent specification by humans.

In the service operation layer, this example shows three distinct intent management function. An OSS intent manager receives all intent for service operation coming from the business operations layer. This intent manager then decides the service operation processes needed. It breaks down the received goals and requirements into suitable instrumental goals. These are then distributed to sub-domains of service operation as intent. Here two sub-domains are shown: Orchestration and Network management. However, this is only meant to be an example of the principles of Intent-driven operation. In a real-world autonomous network, there are more and potentially different domains and layers from the ones shown here.

Resource operation consists of multiple autonomous domains. Using the intent mechanism, it is possible that service operation interacts with each autonomous domain through intent. Typically, intent management functions within service operation would decide what goals and requirements each autonomous domain needs to fulfill. These are the instrumental goals of service operation. These goals are reflected in a set of distinct, but coordinated intent objects, each targeting the intent handler within an autonomous domain.

Please note that intent is not only used between the layers of the Autonomous Networks Framework, but also between autonomous domains, within the layers. Please also note that in this respect autonomous domains do not only subdivide resource operation, but also separate responsibility scopes of intent-driven operation within Service and Business operation layers.

# 8. Intent life-cycle

Intents are distinct knowledge objects with separate life-cycle. This life-cycle is managed by intent management functions. Each instance of an intent management function can assume the following lifecycle management roles:

**Intent Owner**
The intent owner is the origin of intent. It has created the intent object and it is responsible to manage its lifecycle. This includes changing the intent content if needed and finally removing the intent object. Only an intent management function in the role of an intent owner is allowed to create, modify or remove the intent.

**Intent Handler**
The intent handler receives an intent object and operates the domain it is responsible for accordingly. Intent handlers do not modify intent, but they can reject it. However, once accepted they are obliged to fulfill the requirements and goals as well as possible based on the resources and solutions it has available. Intent handlers report back to the intent owners about the handling status and success.

Every intent object has exactly one owner and one handler. The relationship between multiple intent owners and handlers is discussed in further detail in IG1253C.

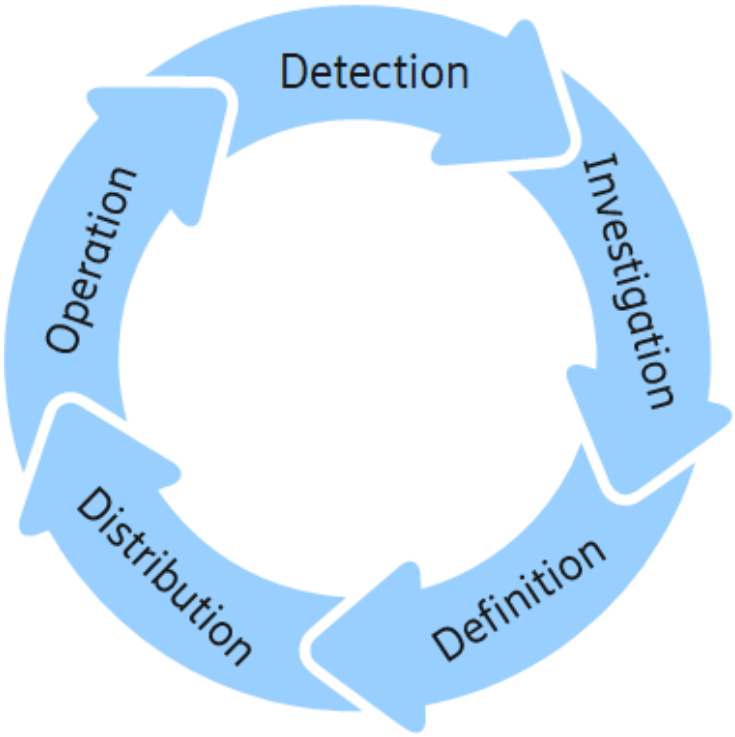


**Figure 8-1: Intent lifecycle phases**

The intent lifecycle consists of the following phases:

**Detection:**
In the detection phase the intent owner identifies if there is a need to define new or change/remove existing intent to set requirements, goals, constraints. An intent management function has its own terminal goals to fulfill. It would break its terminal goals down into a suitable set of detailed instrumental goals. Typically, these instrumental goals need to be fulfilled by other functions and domains and therefore they need to be not only defined but distributed to suitable handlers throughout the autonomous system. This is what the intent owner is doing using intent. In the detection phase the intent owner can react to changes in its own terminal goals or to changes in the fulfillment in its instrumental goals. In this respect the intent owner will need to collect information about the goals' fulfillment. Intent reports coming from handlers are one source for this information. Through intent reports the intent owner is able to react on intent handling success. In any case it is the task of an intent owner to assure the fulfillment of its terminal goals and the first step is to detect if any changes are needed in its instrumental goals and therefore in the intent objects it owns.

**Investigation:**
in the investigation phase the intent owner finds out what intents are feasible. This has two aspects: first, it needs to find suitable intent handlers that have the right domain responsibilities and support the intent information the owner wants to define. Intent handler capability management and detection would be used for this process.

The other aspect of investigation would be finding out if the wanted intent is realistic. This means, if the intent handler would be able to successfully reach the wanted goals and meet the requirements. This depends on the current resource situation and state of the system and can vary over time. Typically, the feasibility of intent is done through a guided negotiation process between the intent handler and intent owner. The owner can explore what the handling result of a wanted intent would be, what would be the best result the handler can achieve, or what would be the most challenging requirements, the aspiring intent handler can offer to fulfill.

Feasibility checks and negotiation can become a challenging task for the intent handler. It might involve nested requests to further intent handlers, advanced prediction models or a combination of both.

**Definition:**
At the end of the investigation phase the intent owner knows what is possible and which handlers can be used. By combining this information with the needs that were identified in detection, the intent owner can now decide and plan all needed intents. In the definition phase the intent owner formulates the intent it needs to use, and it creates the respective intent objects.

**Distribution:**
In the distribution phase the intent owner contacts an intent handler in order to send a new intent or modify or change an existing one. This way the intent owner acts on the plan it has made in definition phase. In this phase an intent management function becomes intent handler by receiving new intent. The intent handler decides if it can accept the intent. If not, it would send a report with the rejection reason back to the owner. While this finishes the life-cycle of this particular intent object, the intent owner can start over with detection to create a new plan. If the intent handler accepts the intent, it starts operating based on it.

**Operation:**
Each intent an intent management function handles constitutes yet another set of goals and requirements to be considered in its decisions and actions. Intent handlers operate their domain of responsibility according to the given intent. They also report back to the owner about status and success while continuously reacting to intent fulfillment threads. Intent reports would be evaluated by the intent owner as part of its detection process, which leads to the next iteration of the intent life cycle.

# 9. RACI of intent and intent handling

We use a responsibility assignment matrix [raci] to discuss the roles and responsibilities of intent-driven operation. We propose to follow the RACI model and its role distinction for tasks. While RACI models were originally designed to describe tasks and roles in human executed processes, we extend and apply the same model to describe entities within an autonomous network. In this chapter we focus on the tasks and entities involved in intent-driven operation.

**R = Responsible (also recommended)**
Those who do the work to complete the task. There is at least one role with a participation type of responsible, although others can be delegated to assist in the work required.

**A = Accountable (also approver or final approving authority)**
The one ultimately answerable for the correct and thorough completion of the deliverable or task, the one who ensures the prerequisites of the task are met and who delegates the work to those responsible. In other words, an accountable must sign off (approve) work that responsible provides. There must be only one accountable specified for each task or deliverable.

Accountability is linked to liability. A party that is accountable for the actions done to perform a task is liable for the effects and consequences these actions have. This means a party can only be accountable if it can be liable in the legal sense. This is a necessary property of any party to be considered accountable according to the RACI model. Legal persons such as corporations can therefore be accountable. Individual persons can be accountable only if they are in a position that implies liability. Some management positions can have this property of the person being personally liable. Persons in positions that do not directly imply liability for their actions can therefore not be accountable in the sense of a RACI model. They are typically allocated in a hierarchy with a person or organization above them, which is liable.

Machines and autonomous systems are never legally liable and can therefore can never be accountable for a task. Accountable entities assign responsibilities for tasks, and they might have assigned responsibility to other persons or to an autonomous system. This means, that responsibility can be handed down in a hierarchy and shared when a task is subdivided into sub-tasks and therefore the responsibility for the task is sub-divided into a set of responsibilities for the sub-tasks. These responsibilities are assigned to one or multiple individual entities. Accountability always stays with the person or organization that is liable and has oversight over all delegated responsibilities no matter if they were delegated to other persons or autonomous systems.

**C = Consulted (sometimes consultant or counsel)**
Those whose opinions are sought, typically subject-matter experts; and with whom there is two-way communication.

In a technical infrastructure one entity with the responsibility for a task might utilize the services of other entities as needed. This can be an information or data service delivering input needed to perform the task. Also, policies can be consulted as part of a decision process the responsible entity performs. Furthermore, a person can be consulted by an autonomous system as part of the process through a frontend.

In this respect, a consultation activity can be broken down into subtasks and therefore implies a set of sub-tasks with distinct responsibilities. For example, the insights of an analytics function can be important for performing a task. The entity with the responsibility of the task would therefore consult the analytics function about this insight. On the other hand, creating the insight is a process that can be broken down

into a set of tasks with associated responsibilities. In this respect it matters for the RACI analysis what level and scope of functional decomposition is chosen to describe the system. In this chapter the analysis is done to describe intent management and the main tasks associated with intent owners and intent handlers.

**I = Informed (also informee)**
Those who are kept up-to-date on progress, often only on completion of the task or deliverable; and with whom there is just one-way communication.

When assigning RACI roles to the technical entities involved in intent handling, informing is interpreted as sending information on the results of a task or sub-task to a party whose own responsibilities depend on it. For example, when an intent handler has done actions towards intent fulfillment, it will inform the intent owner about the results.

A RACI model is typically used to describe and discuss the roles of human actors with respect to tasks in a process. Here we propose to extent the scope of RACI models to also include automated entities such as the intent management function in the role of intent owners and/or intent handler. In this respect it is important to capture the relationship between the human workforce and autonomous systems, how they share RACI roles and how they interact with each other in taking these roles.

The beneficiary of intent is the person or entity (legal person) whose concerns and needs are addressed by the intent. They are the entities benefiting from the intent. There are many parties that can be considered to be beneficiaries, for example:

- **Customers of the service provider**
  Customers and associated end users benefit, because intent expresses and communicates their needs with respect to services that need to be delivered and their detailed agreed characteristics. This is typically legally controlled through contracts including SLA or frame agreements. Their details are expressed as intent and combined with intent directly provided through self-service portals.
- **Service provider shareholders**
  The owners of network operators and service providers are beneficiaries of the intent the service providers use for steering the operation of their infrastructure and network. They benefit from services being delivered with cost optimized resource usage while receiving compensation for fulfilling the contractual obligations. Intent is in this respect a tool to steer the more or less automated systems into the direction of a preferential business result and return of investment.
- **Legislator and Regulator**
  For example a market regulator might use intent towards the operator to enforce legal requirements. Compliance to regulation constitutes their benefit.
- **Other parties or interest groups**

  Further entities can become beneficiaries if they have a concern to be satisfied by the operator. This concern might then either be expressed explicitly as intent or it is communicated in a different way, but influences the intents being used within operation.

The service provider or operator is the central acting party for intent-driven operation. It receives intent from beneficiaries. These intents are external if they are coming from a party outside the administrative domain of the service provider.

A service provider with an autonomous or semi-autonomous network employs a human workforce in combination with automated or autonomous systems. Intent is used to communicate the concerns of the operator to its partly human partly autonomous workforce. This intent is internal. It is primarily expressing the service provider's concerns and needs. One concern is typically to fulfill the needs of external beneficiaries. Consequently, internal intent reflects external intent. Additionally, internal intent distributes inherent concerns of the operator, such as its business goals, strategies and policies.

In this environment of business and legal relationships, the service provider is accountable for all tasks within the administrative domain it controls. This implies that the service provider is liable for the effects and consequences caused by the actions being done by any entity within this domain. These actions are executed by the personnel the service provider employs or the autonomous systems utilized and assigned tasks. The service provider is therefore accountable for Intent-driven operation with all involved actions and effects they might have on beneficiaries. This is the case for all external intent received and accepted by the service provider as well as for all internal intent the service provider uses to steer the underlying infrastructure. The service provider is accountable for all positive and negative outcomes including also collateral and unintended side effects.

In this respect the service provider must trust that the intents received by external parties are correct and complete and therefore reflect the needs of the beneficiaries accurately. The beneficiaries take the role of intent owners in this relationship while the service provider is the intent handler. The beneficiaries are therefore accountable and responsible for correctly performing the tasks of an intent owner. This means they are accountable for the tasks involved in defining the intents and therefore liable if the intent they send is not correct. Nevertheless, the service provider is accountable also for concerns that are not explicitly stated by intent. For example, legal compliance always applies and can also not be overruled by externally received intent.

In terms of the RACI model, the service provider's personnel is responsible for tasks of intent-driven operation including its sub-tasks. We propose to extent this notion of responsibility and therefore the scope of the RACI model to autonomous technical systems. An autonomous system can therefore be directly the responsible entity for a set of tasks. This means, while accountability for tasks is always assigned to a person or legal person, the responsibility can be assigned to human individuals and teams as well as autonomous systems.

Intent primarily targets the behavior of autonomous systems operating a technical infrastructure. It impacts the utilization of typically common and limited resources. This means that actions done for the task to fulfill the intents associated with one beneficiary can impact the outcomes for other beneficiaries. The accountable party is liable for these side effects. It is therefore an important aspect for the responsible parties to consider negative side effects when executing their tasks. Intent defined by the accountable entity can be used to steer this process.

An autonomous system is usually under human supervision. This refers to a person or team responsible to monitor the autonomous system and ready to step in. This might be needed, if the autonomous system for example does not fulfill intents, takes too risky decisions or misses out on optimization opportunities. The human team can apply a range of actions. They might re-configure the autonomous system, potentially by adding intent to shift its behavior. Artifacts such as policies, applications, models or data might be missing although needed or the use cases the system is asked to operate. In this respect ML models might get out of sync with reality and data scientist need to step in and tune or re-train the model with new data. This would be enabled by well-defined and efficient AI governance processes including monitoring and

explainability. Ultimately the human team might also completely take over and operate manually if no other action works.

The autonomous system and the human team monitoring and maintaining it can be seen as a unit with respect to intent-driven operation. They share the responsibility for a set of tasks, but with variable allocation between the autonomous systems and the human technician. In normal operation the autonomous system is the sole responsible entity for operational tasks, while the human concentrates on monitoring correct operation. In exceptional situations, for example if the autonomous system fails in its tasks or if it requires assistance and approval, the split of responsibilities shifts. The autonomous system might escalate tasks to the human, or the human might seize responsibility for some tasks. In extreme cases the human technician might choose to completely take over all operational tasks if necessary.

Within an intent-driven autonomous system the intent management function in the role of intent handler as introduced in chapter 8 is the responsible entity for intent-driven autonomous operation. It receives intent and it involves other functions in the operation. It is also responsible for performing the intent reporting task.

## 9.1. RACI for intent lifecycle management tasks

intent-driven operation is realized by instances of the intent management function by performing intent lifecycle management through the intent handling management service (aka. intent interface). This chapter discusses the classification of major tasks involved in intent lifecycle management and assignment of RACI roles.

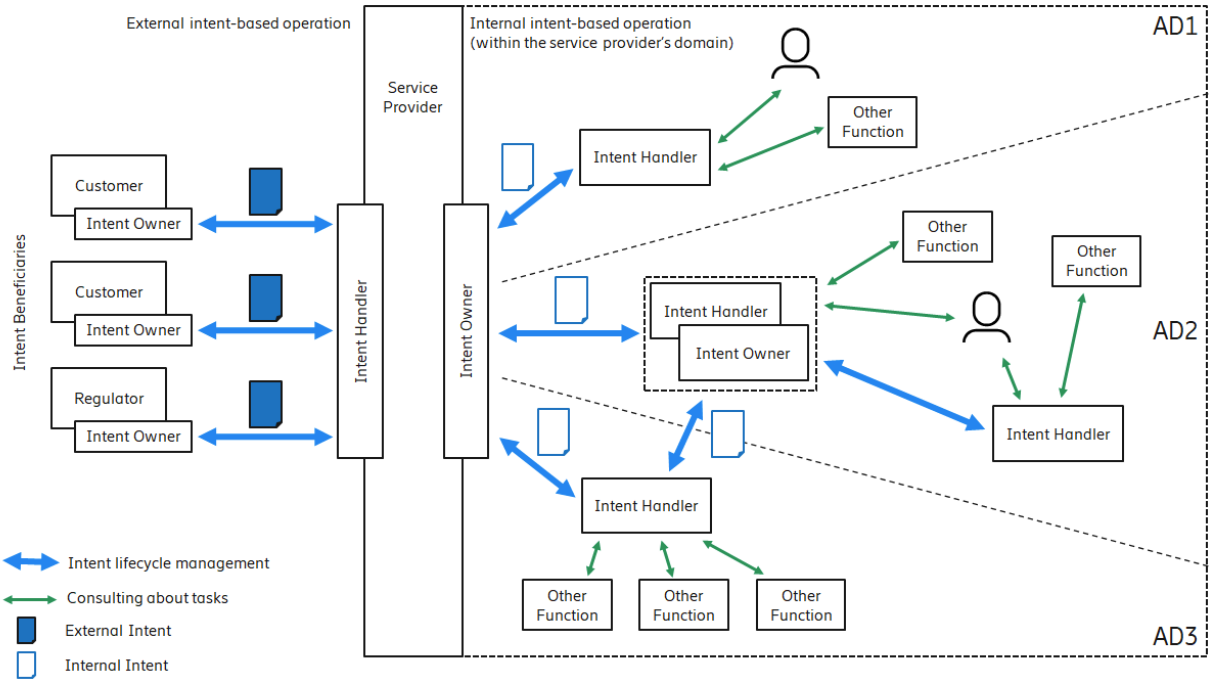


**Figure 9-1: Parties involved in management of internal and external intent.**

Figure 9-1 shows the parties involved in tasks for intent-driven operation. Beneficiaries build a relationship with the service provider by acting as external intent owners considering the service provider as intent handler. The overall administrative domain of the service provider is subdivided into autonomous domains including autonomous multiple layers of intent usage. All tasks involved in intent-driven operation through intent lifecycle management are distributed to intent management functions in intent owner and intent handler roles. The tasks involved and associated assignment of RACI roles are shown in Table 1.

Humans and other functions than intent management functions can be involved in the operation processes, but only as consulted or informed parties. For example, in usual operation humans are only in a consulting role. This usually refers to the personnel of the service provider using an autonomous system to operate and a human team to supervise the system. The autonomous system might choose to escalate and seek assistance from them when needed. Only in exceptional cases, such as failure of autonomous operation the responsibility for tasks in intent management would be reassigned to humans. Table 1 discusses normal operation in which operational tasks and in particular the tasks of intent management are assigned to the intent management function.

Humans also monitor the autonomous system. Tasks associated with monitoring are not part of intent handling and therefore not in scope of this discussion.

**Table 1: RACI assessment of intent-driven operation**

| Intent LCM phase | Task | Human | Intent owner | Intent handler | Other management functions |
|---|---|---|---|---|---|
| Detection | | C | R | C | C |
| | Monitor operational state (intent handlers and infrastructure) | - | R | C | C |
| | Identify the need to change intent | C | R | - | - |
| Investigation | | - | R | C | C |
| | Investigate intent options | - | R | C | C |
| | Feasibility assessment | - | I | R | C |
| Definition | | - | R | - | - |
| | Decide which intent details to use | - | R | - | - |
| Distribution | | - | R | I | I,C |
| | Register intent handling capability profile | - | - | R | I |
| | Select intent handler | - | R | - | C |

| Intent LCM phase | Task | Human | Intent owner | Intent handler | Other management functions |
|---|---|---|---|---|---|
| | Communicate intent | - | R | I | - |
| | Assess and accept intent | - | I | R | C |
| Operation | | C | C,I | R | C |
| | Monitor operational state (underlying Infrastructure) | - | - | R | C |
| | Detect deviations | - | - | R | C |
| | Plan action | C | C | R | C |
| | Execute action plan | - | - | R | C |
| | Report handling status | - | I | R | C |

R: Responsible, C: Consulted, I: Informed, -: not involved

intent-driven operation within an autonomous network is driven by the intent management function. The intent lifecycle determines the distinct tasks involved in the operation. These tasks are split mainly between an intent management function in the role of intent owner and an instance in the role of intent handler. The intent lifecycle is introduced and discussed in Chapter 8.

Next to the intent management functions there are other management functions participating in the operation and also humans might get involved. Table 1 summarizes the assessment or roles according to the RACI model.

Please note, that this assessment is done from the perspective of a single intent instance. In general, and with multiple intents involved intent management function can assume multiple roles. For example, an intent owner for one intent can be the handler of other intents. In these scenarios these other intents the intent manager instance handles will play a major role in the decisions about the intent it owns. A separate assignment of RACI roles would apply to each distinct intent object separately.

**Detection phase**

The responsibility for tasks in the detection phase lies with the intent owner.
Intent handlers contribute through intent reporting. Other management functions also contribute if needed for example with analytics insights and measurement of KPI. This means intent handlers and other management.

Functions are consulted in the task of monitoring the system state.
The intent owner is responsible to identify if new intent or changes to existing intent is needed. The intent owner is ideally making this decision autonomously, buy it has the option to consult with humans if needed.

**Investigation phase**

The intent owner drives the investigation and is therefore the responsible entity. This investigation is mainly about asking intent handlers if they can successfully handle the wanted intent and what would be the result if this intent would be given to the handler. The owner explores options and consults intent handlers. One significant task in this

process is the assessment of potential intent fulfillment success within an intent handling scope and with the available resources. The intent handler is responsible for this task. This task is enabling the intent handler to be consulted in the overall investigation phase. The intent owner will be informed about the results. Other management functions might get consulted by the intent handler if needed in the feasibility assessment.

**Definition phase**

Based on the information gathered in detection and investigation, the intent owner would select one of the investigated options and creates intent objects accordingly. This phase and all its associated tasks are in the responsibility of the intent owner.

**Distribution phase**

In the distribution phase the intent owner is responsible for the task of communicating the intent created in the definition phase to the intent handler. Consequently, the intent handler is an informed party. In return the intent handler informs the intent owner if it accepts the intent. The intent handlers is responsible to do so and to perform all necessary assessment and evaluation tasks potentially consulting with other management functions.

A key task in the distribution process is the selection of a suitable intent handler based on intent handling profile. This is a process enabled by the intent handler registry. It is the responsibility of each intent handler to inform the intent handler registry about its intent handling capability profile. The intent handler registry is consulted by the intent owner when performing intent handler selection. Intent handler capability profiles and the intent handler registry are introduced and discussed in more detail in Chapter 12.

**Operation phase**

In this phase the underlying infrastructure is operated to fulfill the intent. The tasks of the operation phase are therefore primarily in the responsibility of the intent handler. The intent handler needs to monitor the state of the underlying infrastructure. It is gathering all needed information by consulting with other management functions, such as inventories, data management systems and analytics functions.

The intent handler then needs to identify the reason to act derived from a discrepancy between the monitored state of the system and the wanted state defined by intent. Other management function might be consulted for this task.

The intent handler would plan suitable actions to improve the fulfillment of intent. It would consult with other management functions if needed. They can help to propose actions, predict their potential impact, or evaluate their feasibility. If needed, the intent handler might also consult with the intent owner or directly with humans about the preference of alternative action plans.

Ultimately the intent handler has decided how to act and will execute the actions. The intent handler is responsible for the execution as a whole. This might involve multiple sub-tasks other management functions are responsible for. With respect to the overall responsibility of the intent handler these other management functions are therefore informed or consulted depending on the details of the actions involved.

The intent handler is also responsible to report intent handling status and inform the intent owner about it.

### 9.1.1. RACI of intent handling capability management

Instances of the Intent management function participate in tasks for intent handling capability management. The analysis of these tasks according to RACI is shown in Table 2.

**Table 2: RACI assessment of intent handling capability management**

| Task | Human | intent owner | Intent handler | Intent handler registry |
|---|---|---|---|---|
| Register intent handling capability profile | - | - | R | I |
| Select intent handler | - | R | - | C |
| | | | | |

R: Responsible, C: Consulted, I: Informed, -: not involved

Intent managers are responsible to register themselves in the Intent manager registry described in Chapter 12 and IG1253D. The intent manager registry therefore becomes the informed party.

The intent owner needs to identify and select suitable intent handlers as part of its tasks and role within intent lifecycle management. The intent owner is responsible for this task and the intent manager registry is consulted about the intent handler capability profiles. These tasks are performed over the intent manager discovery interface described in Chapter 12 and IG1253D.

# 10.Intent interface

The intent handling interface is the means of communication between two intent management functions. One of them is in the intent handler role and the other in the intent owner role. This interface and its procedures are closely related to the phases and tasks within intent lifecycle management.

This chapter provides a summary of the intent interface. IG1253C contains a more in depth discussion of the interface with detailed examples about the communication procedures. This proposal is inspired by the introduction of interfaces through management services as described by 3GPP in [28.812].

The intent handler interface is independent of use-cases, application domains and system layers. It is primarily concerned with managing the lifecycle of intent objects and related intent reports. All domain specific information would be encapsulated within the intent and subject to the modeling of intent. This domain independence immediately means that an implementation of the intent handling interface can be re-used for every intent management function irrespective of its intent handling scope.

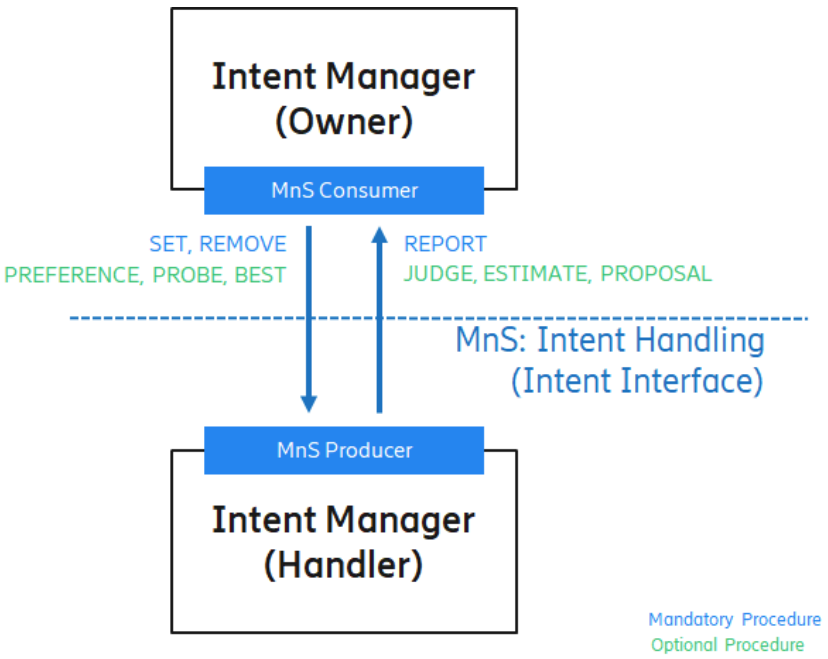


**Figure 10-1: The intent handling management service and interface**

Figure 10-1 shows the intent interface modeled as management service. More specifically, we introduce the intent handling management service. Consumers and providers of the intent handling MnS are both instances of an intent management function. The producer is an intent management function in the role of intent handler, while the consumer is an intent management function in the role of owner. These roles are assumed on a per intent basis. This means the same instance of an intent management function can be the producer of the intent handling MnS for some intent, while it is the consumer for other intent. It is however never both for any distinct intent object.

The intent handling MnS defines a couple of procedures for communication. Mandatory procedures need to be implemented by any intent management function. They represent the bare minimum of communication needs between intent handlers and owners.

Optional procedures and the respective communication procedures address advanced features such as intent negotiation and owner/handler collaboration on prioritization. This is optional, because supporting them would require advanced capabilities of intent management functions, such as predictive and speculative models. Simple intent management function implementations would typically not have these capabilities due to cost of complexity or because they are not needed for the responsibility domain of an intent management function.

**Mandatory Procedures:**
These procedures need to be supported by every intent management function.

SET:
This procedure is used by an intent owner to communicate the new or modified intent to an intent handler.
This is how the intent owner communicates the needed intent. The handler would reply with an initial report. The handler can accept or reject the intent.

REMOVE:
This procedure is used to remove an intent object. This is how the owner of the intent object retires (delete) intent that is not needed anymore and informs the handler. The handler will send a final report in return and remove the intent from the set of requirements that determines its operational actions and decisions.

REPORT:
This procedure is used to communicate intent reports. It is initiated by intent handlers once they are required to report the intent handling status and success back to the owner of the intent.

**Optional Procedures:**
These procedures can be implemented if needed. They provide features for negotiating intent and for collaborative optimization of decisions and actions. This means, implementing optional procedures can lead to higher levels of autonomy and optimized operation on the intent control loop. An intent management function can use the mechanism of intent handler capability management to announce its range of support.

JUDGE, PREFERENCE:
This procedure allows a collaborative evaluation of proposed solutions. This is part of the operation phase within the intent life cycle. The handler asks the owner to judge alternative outcomes corresponding to alternative solution strategies and actions. This is done before these actions are executed. This way the intent handler achieves a better understanding of the intent owner's needs and priorities that were not apparent from the intent.

The intent handler needs to formulate hypothetical outcomes for alternative actions as base for a judgment decision. This requires predictive capabilities able to estimate the effect of action on the system state.

PROBE, ESTIMATE:
This procedure allows exploring the potential handling results of an intent. The process is typically part of the investigation phase within the intent life cycle. It is initiated by the intent owner. It involves sending an intent object to the handler, asking the handler to not consider it in actual operation decisions and actions. The handler shall rather just estimate what the outcome would be, if the intent would be really set by the owner. The

handler would deliver its assessment in the form of an intent report. This procedure can be used to establish a process of negotiation between the intent owner and handler. Providing a good estimate requires predictive capabilities within the intent handler. Predictive machine learned models and digital twins might be used to gain this capability.

BEST, PROPOSAL:
By using this procedure, an intent owner asks an intent handler for the best intent configuration it can successfully handle. This refers to the most severe requirement the handler would be able to successfully comply to. The owner guides the handler by pointing at expectations to focus on with respect to required/prioritized value ranges. Guidance by the intent owner is important, because it understands priorities and importance of one requirement over another one. The asked intent handler does not have this knowledge because it is derived from concerns and needs of another domain. Therefore, it cannot decide itself what proposal would be sensible without being told what to focus on.

This procedure can be used as part of the owner and handler negotiation within the investigation phase of the intent life cycle. It is also possible to use this procedure for asking the intent handler to provide proposals also for intents that are in operation. This means the intent owner would not only receive information about the current status of intent handling in an intent report. It would also receive proposals for the most sever requirements the intent handler considers to be possible.

# 11.Intent management scope

An instance of the Intent management function operates a system according to a clearly assigned responsibility for task. This refers to the set of responsibilities according to the RACI model as described in Chapter 9. In this respect implementations of intent management functions are not generic but specialized in the tasks they are responsible for. The scope of task responsibility typically matches clearly defined sub-system, ANF layers or autonomous domain borders. This means every sub-system or domain within an autonomous network has a unique instance of an intent management function with the responsibility to fulfill the intents targeting this domain. The range of tasks and borders of responsibilities are referred to as intent management scope.

IG1253D contains a more detailed discussion of intent management scopes a collection of proposed intent handling scopes. This chapter demonstrates the concept. It will be revised and extended in future releases based on detailed use case studies and potentially cross SDO collaboration. In this respect the scoping of intent handling also plays an important role in the governance of work executed across SDOs and standardization work groups. The scope of a domain specific intent standard proposal ideally targets one or several intent handling scopes. It is in this respect explicitly allowed to define new intent handling scopes as needed. And it is encouraged to contribute them into the list and documentation of intent handling scopes in IG1253D. This would ensure that overlaps in scopes are discovered that they can be avoided.

# 12.Intent manager capability management

Depending on their scope of operation and responsibilities, instances of the intent management functions need domain specific implementations. This means intent management functions have unique capabilities. Aspects of intent manager capability are:

- Intent manager scope: This describes the range of responsibility for operational tasks within a system domain or sub-system. By associating a defined scope with an intent management function, this intent manager claims that it is the responsible entity for all intent targeting this scope.
- Intent interface support: This refers to the optional intent interface procedures supported by this intent management function. This means that this intent management function has implemented also some of the more advanced and challenging processes implied by the optional interface procedures.
- Intent notation format: Intent and intent reports are represented as knowledge graphs. For the transmission over an interface these graphs are serialized and there are various notation formats available for this purpose, for example TURTLE, XML or JSON-LD. An intent management function might have implemented support for all of them, a subset of them or it might even support further alternatives or proprietary formats.
- Intent expressiveness: This refers to the intent extension and intent information models understood by an intent management function when used within intent and intent reports. The intent management function understands any intent if it is built using a model federation from this set of constituent. This does not imply, that it will always be successful in fulfilling the intent, but it understands the meaning of the requirements the intent carries.

## 12.1. Intent manager profile

An intent manager profile is an information object that describes these capabilities offered by an instance of an intent management function.

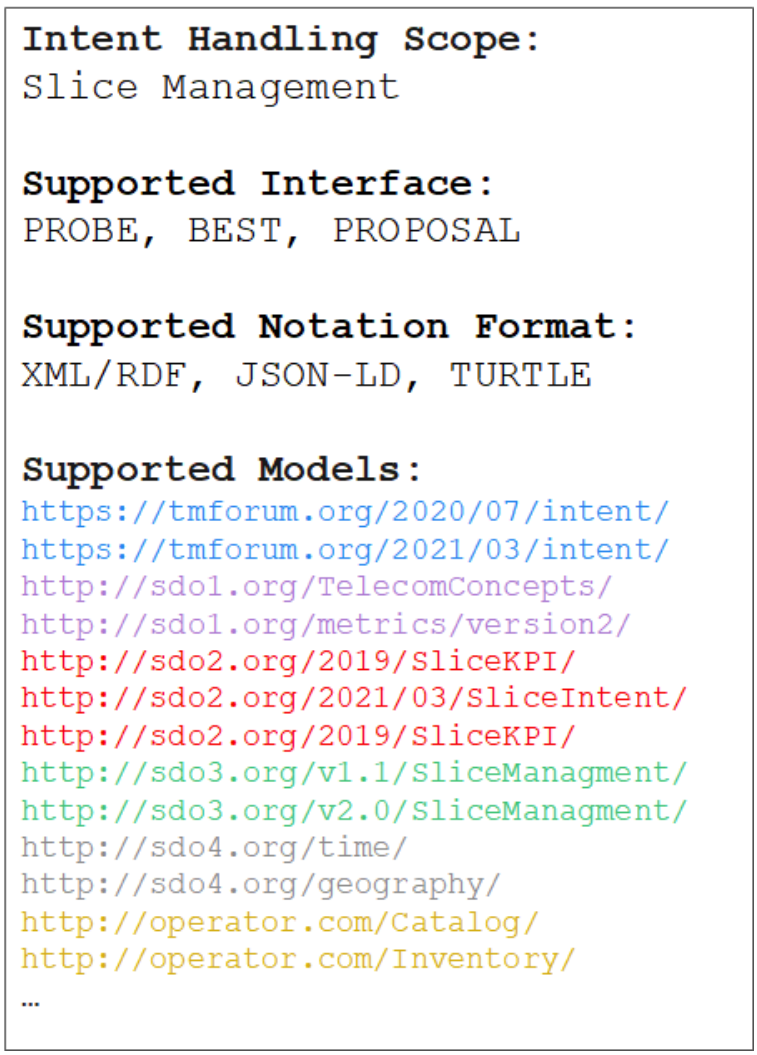


**Figure 12-1: Example profile of an intent manager.**

Figure 12-1 shows an example intent manager profile. In this example the intent management function has the intent handling scope of "Slice Management". Furthermore, it supports the PROBE, BEST and PROPOSAL optional procedures on the intent interface and intent and intent reports can be transmitted encoded in XML, JSON-LD and TURTLE notation.

Furthermore, the profile lists the supported models for intent and intent report expression. The intent manager described by this intent manager profile supports and understands intent and intent reports expressed using these models.

This capability profile information is important for multiple tasks within the intent life cycle. It allows an intent owner to identify the intent handler instance that is responsible for the domain targeted by the intent it wants to set. The profile also determines what vocabulary is available in the communication with that targeted intent handler. Furthermore, the profile provides information about which interface procedures for intent feasibility study and requirement negotiation would be available to use in the investigation phase of the intent life cycle.

Please note, that the list of supported models also contains the intent common model, although it is considered to be mandatory for every intent management function to implement. However, there might be multiple versions of the model and the intent handler profile allows defining, which of them are supported.

More details on intent manager capability profiles and the model for encoding them are described in IG1253D.

## 12.2. Intent handler registration and discovery

The information contained in the intent manager capability profile of one intent manager needs to be considered by other intent managers. They require the information about each other's capability to collaborate effectively in intent life-cycle management and intent-driven control loops. In this chapter we introduce a mechanism how the intent manager capability profiles are distributed. More details and a proposed model for formulating the profile is provided in IG1253D.

Intent management functions register themselves with their intent handler profile at the intent manager registry. This is done through the intent manager registration interface exposed by the intent manager registry. In this respect it is the responsibility of each intent management function to keep its profile information up to date. This can be needed for example if dynamically deployed artifacts such as policies or machine learned models introduce or remove capabilities.

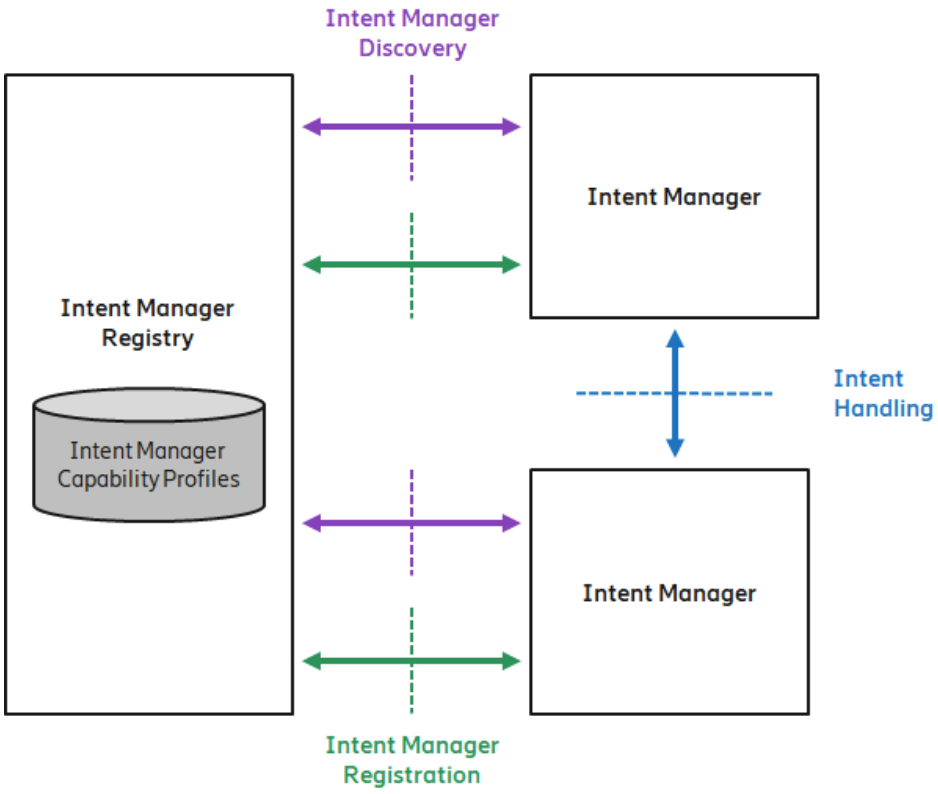


**Figure 12-2: Intent manager registration and discovery**

Figure 12-2 shows the interfaces involved in intent handling, intent handler registration and intent handler discovery. Within an administrative domain, such as an operator's network, there will usually be one intent management registry. This is only logically centralized and distributed deployments for redundancy and performance reasons can be chosen.

If intent is used directly across administrative boundaries, for example between operators and their customers a common intent manager registry can be used. Alternatively, both parties can grant access to their respective intent managers and expose the profiles of the intent managers that shall be visible. For example, the operator might expose the intent manager for order management to their customers.

The intent manager registry exposes an interface for discovery. It allows to query the intent handler registry. This way intent owners can identify suitable intent handlers for the intents they want to create.

It would also be possible to expose different profiles internally and externally. This can be needed if the operator would allow customers to formulate intent about service orders, while additional intent about the operator's policies and goals regarding contract and SLA acceptance can only be defined by the operator. The detailed authorization and profile publishing mechanisms for these use cases are future work.

# 13.Modeling of intent objects and reports

## 13.1. The nature and use of intent models

Intents were defined as being the requirements, goals and constraints that reflect needs and communicate them to the autonomous system. This is knowledge created by an intent management function and sent to another intent management function to be considered as requirements in its autonomous operation. While intent might be based on the intentions and needs of a human or an organization, intent objects are practically exchanged between machines. They need to arrive at exactly the same interpretation of the intent content, which is achieved by standardized models. This means, we understand intent modeling as a challenge for knowledge modeling with comprehensively defined vocabulary and semantics.

Intent modelling is not done to communicate to a human developer what needs to be implemented. This type of models are only directly used by a human developer and the implemented system complies to the model, because the developer has implemented it accordingly. Interface information and data models in API design are typically used this way. UML is by far the most popular modelling environment and language for this style of model use and it provides sophisticated tooling and expressiveness for this purpose.

Intent models are used differently. They allow creating information objects that are directly consumed and worked with by a machine. This means the developer is not a necessary middleman anymore to transform the model into implemented logic. The machine will directly reason about the modeled objects and therefore makes online use of the model and all its semantics definitions directly. This means, intent management involves a machine reasoning about the knowledge conveyed through the intent model within an environment of knowledge about the operated systems and infrastructure. This means intent models must have formal semantics that allow machine reasoning and logic inference. Modeling with formally defined ontologies would enable this capability. This approach is also essential for building more advanced self-adapting systems and that is a key characteristic of higher levels of autonomy. A popular and widely used standard for these modeling challenges is the Resource Description Framework (RDF) [rdf] by the World Wide Web Consortium (W3C) [w3c].

Individual intents are knowledge objects represented and expressed in the form of ontology graphs. This means that any language able to specify ontology graphs is in principle suitable for expressing intent. We recommend however to base the intent modeling on the Resource Descriptor Framework (RDF) [rdf] family of standards including the related RDF Schema (RDFS) and the Web Ontology Language (OWL). These are actively developed standards with broad application in knowledge management use cases. They were developed to formally express knowledge within machines and enable automated inference from the models.

Intent-driven operation as proposed by IG1253 is based on two central knowledge objects: intent and intent report. The discussion regarding modeling techniques, languages and base standards are valid for both.

## 13.2. Expressiveness of Intent

Intent modeling defines the expressiveness of intent. It introduces vocabulary and semantics needed to express and encode the knowledge about requirements, goals and constraints that an intent object shall carry. Standardized intent models define

common semantics in order to enable two parties to agree on the meaning of intent and remove ambiguities and divergent interpretation.

The intent model needs to provide sufficient expressiveness to cover all relevant use cases. Autonomous operation is, in general, a multi domain task. Consequently, intent models must provide domain-specific and use case specific expressiveness. For example, intents at a business level might express user requirements as well as goals on financial results. These intents would determine, for example, the outcomes of automated order and contract management, which might include automated contract negotiation. These are the responsibilities of the business layer and the intent used on that layer expresses matching requirements, goals and constraints.

Other domains would have different responsibilities and operational tasks. They deal with different types of resources and therefore require different expressiveness within the intent. For example:

- Concerns of RAN management include coverage, quality of service and availability of services as delivered by the radio network and individual cell sites and resources within it. Intent targeting RAN would need expressiveness for setting requirements about these concerns.
- Network function management deals with deployment of network functions with typical requirements concerning scale and allocation.
- Slice management has the concern to set up and configure network slices to that they deliver the required quality of service.
- Service management is concerned with end-to-end orchestration and assurance and is required to assure an overall satisfactory user experience.

Each of these domains has specific types of requirements. Consequently, intent targeting those specific requirements needs to be based on a model that provides vocabulary to express these specific requirements. It is therefore a central concern of standards for intent modeling to provide comprehensive domain-specific expressiveness. And, because intent is communicated between intent management functions in different domains, intent modeling is a cross-domain task.

While domains have considerable differences there are also many common concerns with respect to model expressiveness to the extent that modelling concepts and vocabulary can be shared. For example, all domains need the expressiveness to define numerical goals based on metrics and KPI defining thresholds, target values and required ranges. The exact metrics that are used are highly domain-specific. For example, BSS might be concerned with cost metrics and high level user experience measures, while autonomous domains in resource management deal with requirements that express performance of distinct resources. All metrics and KPIs have in common is that they are measurable with a numeric value. This means a model that provides the vocabulary for referring to a metric and set a numeric target would directly be useable in all domains and cover a broad range of intent use cases.

Another example is that all domains typically need to express the requirement that something needs to be *delivered*. What this something is varies with the domain. It can for example be a service, a slice, an application, a network function or even an entire network. But this is only a variation in the targeted object or resource and not in the semantics of the requirement that something needs to be delivered. It is possible to express this with generic and domain independent vocabulary. We can conclude from this discussion that intent in various domains are similar with respect to what expressiveness they need from the intent model, although the details can be highly domain-specific.

It should be noted that developing a single model that contains a superset of all domains, application and use case expressiveness is not practical. The size of the model, the diverse competences needed to develop it and the organizational challenge of central governance are problematic. Domain expertise is distributed across projects and work groups within various specialized Standards Defining Organizations (SDO). It is good practice that specialized work-groups define the standards specific to a domain. On the other hand, this practice often leads to significant differences in modeling and interface design even for similar challenges. This ultimately leads to high cost in developing similar interfaces and functions several times.

In intent standardization we have a combination of common modelling challenges and domain-specific ones. A considerable amount of expressiveness is domain-agnostic or domain-independent. Also, the interface for life cycle management and negotiation of intent can be designed in a domain independent and therefore re-useable way. The proposed way to approach this is through using federated models as explained in Chapter 14.

## 13.3. Requirements and Concerns for Intent Modeling

This chapter lists and explains essential requirements for intent modeling. They provide the base for proposals ranging from the choice of modeling standard to the detailed expressiveness. This chapter also explains how the chosen and proposed concepts will meet the requirements

### 13.3.1. Intent is knowledge

Intent carries knowledge about requirements, goals, and constraints, including context and supplementary information. It does this based on suitable vocabulary and semantics. We understand intent modeling as a challenge of expressing abstract knowledge in a way that a machine can draw conclusions from it. This means intent models shall be anchored in an ontology that covers the abstract concepts involved with vocabulary and implied semantics accessible to machine reasoning tools.

We propose to use the Resource Description Framework (RDF) as the base framework for intent modeling. RDF is a popular standard in knowledge modeling and knowledge management applications. It has a proven track record for modelling highly interconnected data and tooling around it is mature. This includes machine reasoning tools that derive inference from the knowledge in automated processes.

### 13.3.2. Ambiguity free semantics

Intent models need to express the semantics of intent so that two intent management functions can agree on the meaning. Ambiguities in the model would lead to interpretations depending on assumptions made in the development of intent managers. Divergent interpretations are hard to avoid if the model for intent expression leaves room for it. This is important, because intent is used on cross domain and cross system interfaces in heterogeneous multi-vendor environments. Even under these circumstances all involved systems must agree on the meaning of intent even though these systems were independently developed. Intent standards must ensure intent expressions are ambiguity-free with comprehensive specifications and formal modeling approaches. Formal models based on ontologies of abstract concepts as well as modularity through federated constituent models are core strengths of RDF, and therefore two reasons to justify its selection for intent modelling.

### 13.3.3. Domain awareness and domain independence

The proposed intent mechanism is intended to work across multiple industry domains and within a single domain across all involved systems and sub-systems. It is therefore not preferable to choose techniques that are only applicable in a subset or even a single application domain. The base for common intent modeling should therefore not contain domain-specific constructs or assumptions. On the other hand, Intent modeling must allow domain specific expressiveness to allow precise semantic mapping of vocabulary for all relevant concerns of an individual domain. Furthermore, the scope of intent-based operation will expand over time into additional domains with their unique requirements.

Many practical modeling tasks for intent are similar across multiple domains. There are considerable similarities in vocabulary and concepts needed in various domains. For example, many domains require the expressiveness to set goals based on KPI thresholds and value ranges. This indicates that vocabulary and semantics defined in generic, domain independent models can cover already many practical modeling tasks.

In conclusion, a *modular* approach to intent modeling is preferred. It would allow combining models defining common expressiveness and generic vocabulary with domain specific models that add domain concepts and specialized vocabulary. Extensions can be made by adding additional models or by building new combinations. We refer to this concept as model federation. RDF uses IRI for globally unique references combined with notations that makes it easy and intuitive to combine vocabulary from different namespaces representing multiple constituent models. In RDF, this kind of model modularity is a basic and commonly used mechanism. This particularly allows to seamlessly combine models created and published independently by multiple organizations.

### 13.3.4. Semantics for automated inference

The receiver of intent is always a machine. And this machine needs to draw conclusions from the intent. This is a machine reasoning task enabled by the semantic mapping of vocabulary within the modeling standard. RDF and the related RDFS and OWL have roots in symbolic AI and are often used in combination with logic reasoners for inference. This means that RDF models allow to create the respective semantics and tools and environments are available for implementing reasoning based processes and applications. This constitutes a proven base for implementing knowledge-centric applications with the capability to dynamically react and self-adapt to changes in the knowledge and in the underlying models. It allows implementation patterns where changes in the knowledge modify the system behavior through machine reasoning and logic inference.

### 13.3.5. Knowledge base and efficient query

Knowledge centric operation is based on intense interaction between a knowledge base and functions that implement reasoning about the knowledge for decision and action taking. The intent handling function is a good example where such an implementation pattern is expected. This means efficiency in the interaction between the function and knowledge will directly impact the performance of intent handling. It is therefore a major concern when choosing the platforms and base components for implementing intent handlers.

RDF based models are linked data graphs, typically managed in graph databases. There are many options including highly scalable corporate grade offers. Interaction with the knowledge and in particular queries with implicit reasoning is typically based on SPARQL [sparql]. It is a standardized and highly efficient query language that is often natively supported by databases that can accommodate the knowledge base.

#### 13.3.6. Efficient serialization and notation

Intent is expressed as ontology graph that need to be transferred between intent management function. This means that efficient serialization and notation formats need to be available.

The RDF infrastructure contains multiple formats and languages for serialization, notation and interchange of models. TURTLE [turtle] was developed specially for this task and it also provides a syntax that is intuitively readable by humans. It is therefore often also used as notation format in modelling. Other serialization formats are RDF/XML [rdfxml] and JSON-LD [jsonld]. The use for further formats and languages such as YAML appears possible, but it is not standardized yet.

#### 13.3.7. Convenient human oversight and monitoring

The modelling of intent and its operation including intent reporting and the related knowledge representation must be understandable by humans if they want to observe the operation. This means intuitive presentation tools and formats should be available.

For RDF models there is the serialization through TURTLE with a syntax that is easy to learn and intuitive to use by technically trained personnel. In addition, there are graphical tools as well as text based tools for model visualization and review. This is however only a basic tool set for inspecting the models and knowledge. More advanced presentation that distinguishes the competence and interests of different users would however require the design of a presentation layer that implements the needed features. More advanced systems could also include techniques for explainable AI and automated system monitoring and escalation.

#### 13.3.8. Competence

Personnel competent to work with the chosen modelling technique should be widely available or it must be possible to easily learn the techniques for somebody who is trained in other modelling approaches. RDF is a very popular choice in knowledge management. This means that competent and experienced personnel is available. Furthermore, the effort to acquire this competence in RDF/RDFS/OWL modeling is on a similar level as with other modelling standards and languages.

#### 13.3.9. Open standards

The base techniques and standards for intent modelling must be open and freely accessible. Ideally it is based on standards defined by an organization that is widely accepted and continues to actively maintain the chosen model standard. Ideally it is also neutral, based on industry collaboration and does not impose prohibitive restrictions on participation and membership. This is the case for RDF, which is standardized, published and maintained by W3C.

### 13.4. Discussion of modeling standards

This chapter compares modeling using the Unified Modeling language UML [uml] and the Web Ontology language (OWL) [owl]. The comparison is done as a base for selecting the modeling standard better suited for the challenges of intent modeling. This discussion follows and cites major findings from the paper "A detailed Comparison of UML and OWL" [umlolw]. It presents a detailed and comprehensive comparison of UML versus OWL from the RDF modeling family.  In this respect the statements about OWL apply to the entire RDF/RDFS/OWL modeling stack. The paper is written with the

explicit goal to facilitate good decisions about which technology to use for which purpose and how to use them together and integrate them.

UML is regarded to be an instance of the Meta Object Facility (MOF) and it is partly a specialization of it. UML is typically used in combination with the Object Constraints Language (OCL) [ocl]. It extends UML with a declarative language for describing rules. In this comparison UML is considered to be combined with OCL.

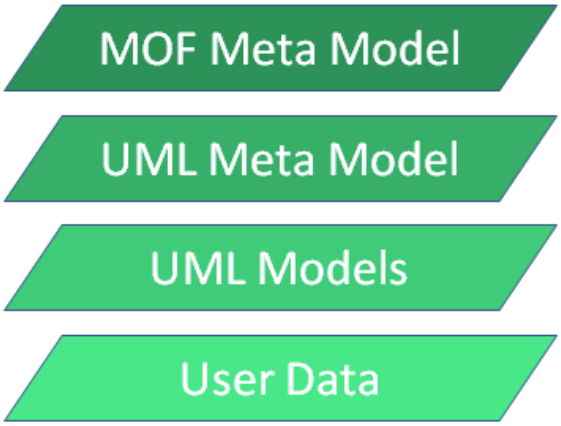


**Figure 13-1: Four layer metamodel architecture of UML**

OWL is the top of a four level modeling framework. It is built on top of RDF and RDF Schema. Each level provides specialization of the level below.

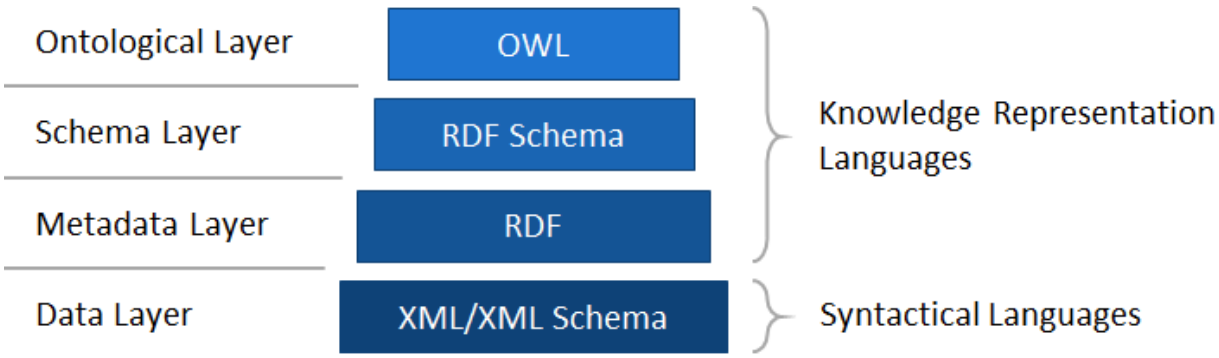


**Figure 13-2: Four layer modeling framework of RDF/OWL**

### 13.4.1. Syntax

With respect to concrete syntax UML has a standardized graphical representation as well as a representation in XML called Metadata Interchange (XMI).

RDF also has an XML notation called RDF/XML [rdfxml]. Furthermore, RDF has a dedicated notation format called Terse RDF Triple Language (TURTL) [turtle], which is often also used directly for editing RDF models due to its intuitive syntax. Another alternative for RDF/OWL models is the JavaScript Object Notation for Linked Data (JSON-LD) [jsonld]. Graphical tools for developing RDF/OWL models exist, but unlike UML the graphical elements are not standardized.

The concrete syntax of UML and RDF are not the same even if they both use XML as concrete notation. Name tags are distinct and synonyms as well as homonyms are possible.

### 13.4.2. Semantics

When considering their semantics, it becomes clear that UML and RDF were devised to fulfill different purposes. RDF/OWL supports the representation of knowledge about a system. UML was developed primarily to support the construction of a software system.

It can be argued that both are similar because the underlying goal is the representation of a system. Both are object-centric and the main component of knowledge representation is an object and its relation to other objects. The relation is known as property, predicate, or association.

Furthermore, both languages allow knowledge to represent an abstraction of the elements of a concrete system. Instead of representing every object explicitly, they allow to represent the common properties of sets of objects and therefore classify objects through their properties.

Both languages have two basic layers of knowledge representation, namely concrete, instance-level information called extensional or assertional knowledge representation, and abstract, type-level information called terminological or intentional knowledge representation. For extensional knowledge representation, the semantic domain is the set of objects and objects relations (object states) under consideration at a given point in time.

Both languages differ in the way knowledge is understood and expressed. This leads to a difference in the role of terminological knowledge and extensional knowledge. Ontology engineering with RDF/OWL uses a terminological knowledge representation approach to classify extensional knowledge and to infer new knowledge from it. If the extensional knowledge contradicts the ontology, it is identified as not satisfying the ontology.

UML system models with a terminological knowledge representation are used in software engineering to represent and constrain the allowed set of system states. A concrete system state (extensional knowledge representation) must satisfy the constraints laid down by the system model to be a legal instance of it.

Both approaches start with extensional knowledge to define a terminology, but while ontology engineering reuses this ontology to apply it to other extensional knowledge to deduce further extensional knowledge, in software development it is used for the construction of a (single) system (i.e., extensional knowledge representation). The consequence is that extensional knowledge in the UML is intended to be dependent on a terminological knowledge representation (i.e., a class model).

The difference between the constructional motivation for UML and the representational motivation behind RDF/OWL is reflected in their different underlying assumptions. These influence the set of available constructs as well as their interpretation.

UML and RDF/OWL make different assumptions regarding the interpretation of language expressions or statements. One of the most fundamental assumptions in knowledge representation is the world assumption. UML interpretation is based on a closed world assumption. This means UML models are always considered to be complete. In contrast, RDF/OWL have an open world assumption and models are interpreted as representing partial knowledge. A difference on world assumption is one of the main reasons why UML and RDF/OWL models cannot simply map into each other.

### 13.4.3. Conclusions and proposals

Both languages were devised for the purpose of knowledge representation and have common roots in object-orientation. This did lead to a high degree of similarity in their syntax. On the other hand, each language has a slightly different purpose that manifests itself in several ways. OWL restricts itself to a set of inference-supporting constructs. This directly enables logical inference through machine reasoning implementations, which is the basis for implementing knowledge-centric adaptive processes. UML, in contrast, is not concerned with automatic deducibility of information from the knowledge and offers an unrestricted constraint language. It additionally focuses on describing implementation related issues that are very useful in the process of designing and implementing a software system, irrelevant for abstract knowledge representation.

Logical inference is a key feature with respect to intent modeling. The receiver of intent is a machine that needs to derive inference directly from the model and by that interpret the intent with respect to additional knowledge about the system state and decide on actions. RDF based models directly support this type of model use. In contrast, UML models would describe how an intent based system needs to be implemented. The receiver of the model is therefore a human developer translating the model into coded logic. Rules and policies. This implementation represents the model semantics and by that compliance to the model is reached. Intent being data instances according to the model would therefore meet an implementation that interprets it as intended.

Both approaches would in principle work to communicate the intent information towards a system that can interpret it correctly and handle it. However, model changes in UML models would require modifying the implementation. This is a coding effort involving the implementation of rules, policies and even the instances of management functions. This means human involvement. In the knowledge-centric and machine reasoning based approach, changes effect primarily the model, and inference based on ontologies can adapt the system behavior. Semantics in the ontology make this work and in many cases human involvement in the system adaptation can be avoided. This is the key for reducing human involvement and by that ultimately reaching higher levels of autonomy.

**The first major reason for proposing RDF in intent modeling is therefore the ontology nature of the models with semantics allowing direct inference through machine reasoning and logic based implementations of intent managers.**

The differences in fundamental assumptions used with OWL and UML lead to different interpretations of common language constructs. In other words, while UML and RDF/OWL share a large set of similar language constructs, these constructs possess different meanings with different mappings in the semantic domain. As a result, it is impossible to directly translate OWL ontologies into UML models and vice versa without the loss or corruption of information. A translation between the models is a major effort that would require rules for domain specific interpretation of models and creating an equivalent model within the boundaries of the domain. Limiting to the semantics that are relevant in the target domain might make this practically doable rather than trying a general purpose translation. However, we do not propose model translation, but using each model paradigm natively for its intended purpose. UML for describing the system implementation and RDF for dynamic knowledge such as intent including ontologies that carry semantics enabling logical inference.

**We propose to use a combination of UML and RDF based models. UML for system architecture and interfaces, RDF for intent and intent report vocabulary and semantics.**

The proposed intent interface is an example of using a combination of UML and RDF modeling. The intent interface information and data model is proposed to be modeled in UML in accordance with API policies and guidelines within the working groups that develop detailed interface and API standards. However, the intent and intent report objects shall be modeled based on the RDF family of models. This means from the perspective of the API information and data models an intent or intent report is basically a string type data element. Its internal structure is not the concern of the interface and API information model. The API is however concerned with the operations of intent life cycle management and intent negotiation. The vocabulary and semantics of the intent and intent report objects are modeled in RDF.

The global identification of model elements is a feature of RDF/OWL that is very attractive for the challenges of intent modeling. Intent is communicated across boundaries of autonomous domains, layers of the operations stack and even across administrative domains. Globally unique identification rather than local contextual interpretation makes it easier to reference objects, resources and data across models and across network functions. These are distributed across management functions, inventories and catalogs with local information models. Intent requires an easy method to reference objects across these functions and their inherent models. Intent needs to refer to objects for defining requirements about them and it uses data objects to specify the detailed requirement.

Furthermore, autonomous networks benefit from modularity in modeling creating domain specific models from a selection of common ontologies and generic vocabulary in combination with domain specific additions. Models are assembled from heterogeneous sources, such as multiple SDOs in combination with operator and vendor specific models. This federation of models is directly supported in RDF/OWL modelling and this support is also based on globally unique identification of objects.

**We propose to create domain specific intent models as a federation of models. RDF provides mechanisms based on its global IRI referencing for simple definition and management of federated models.**

In conclusion, intent can in principle be modeled in UML as well as in RDF. UML has a longer track record in telecommunication and software development, while RDF is rooted in symbolic AI and the modeling technique chosen for knowledge and reasoning centric applications.  Intent modeling is not primarily a software modeling challenge, but rather knowledge engineering. Furthermore, we expect that the implementation of intent management will benefit from logical inference as enabled by RDF based ontologies. We expect that this is a key enabler for reaching higher levels of autonomy through self-adapting capabilities and decision-making based on utility considerations. Despite their similarities, RDF/RDFS/OWL has features that better match with the challenges of intent modeling and intent management. Especially higher levels of zero-touch autonomy will require self-adapting systems. The roots of RDF in symbolic AI enable logic reasoning based implementations, which are a key capability for reaching this goal.

# 14.Model federation

Intent is in general highly domain specific. It must be able to express the requirements, goals and constraints from all domains participating in intent-driven autonomous operation. Domains can be very different which respect to expressiveness and information content of the intent objects they use and understand. BSS for example deals with artifacts and abstract concepts close to customer and business needs, such as contracts, SLA, monetary compensation, customer value and service order. At the other end of the spectrum are domains directly operating the resources of the network, such as radio nodes, cloud datacenters or network slice management. Requirements and goals here are often based on metrics and KPI that reflect the state and performance of distinct resources such as latency and throughput for traffic on a slice or the processing scale of network functions.

Next to technical diversity there are also organizational, administrative and governance aspects. The different layers and domains of an autonomous network or a network in general are subject to focused standardization efforts by multiple specialized organizations, projects and work-groups. This is where domain experts define the architectures, models, interfaces and any standards according to the needs of their domain. In practice this is often complemented with operator or vendor specific extensions.

This diversity means that a single unified modeling effort to cover a globally applicable concept such as intent is impractical. It is therefore not a good practice in the industry. On the other hand, standards are never built in isolation. One standard is built on top others, which contain already defined and agreed more general concepts and specifications. This way the industry stays relatively consistent.

There are concepts in intent based operation that are domain independent. It would be beneficial to keep the common aspects compatible and avoid incompatibilities by addressing the same concerns multiple times in parallel domains. Avoiding incompatible and fractured standardization will have industry-wide cost advantages, because it avoids double implementation of similar functions and interfaces dealing with similar-but different information and data models and interpretation of semantic differences at the cross-domain touch-points.

When introducing a new paradigm such as intent based autonomy it is a challenge to standardize it in a way that allows to cover domain-specific needs adequately and in a modular additive way, while staying conceptually aligned and avoid incompatible parallel work. This is particularly important for intent, as it is a generic concept and operations paradigm. It is only partially domain specific with the explicit goals to make expansion into further domains as smooth as possible.

In intent standardization there are the models for intent expression and the interface for intent life cycle management and requirement negation, where common generic concepts meet domain specific needs.

In the intent interface this is addressed by an interface information model that is separated from the models for formulating intent. This means the intent interface models define semantics of life cycle management and negotiation operations but exclude the expression details of intent and intent reports. IG1253C discusses the interface and its operation procedures in detail.

The models for intent and intent report expression contain generic aspects which are domain independent. For example, all intent have distinct owners and handlers. All intent is formulated as a collection of expectation objects that carry the details about requirements and goals. All intent can furthermore set context for the requirements and provide additional information. Intent reports contain corresponding information about

state of the target system and to what extent it fulfills the intent requirements. All this can be covered with model vocabulary and semantics independent and irrespective of the operational domain. Domain specific vocabulary and semantics would be an extension and specialization of the more generic model definitions. This indicates modular modeling can be very beneficial for intent based autonomous systems.

This modularization can be reached by defining a set of additive models. Models defining domain and use cases specific specialized vocabulary and semantics are doing this by extending a common root model. The intent modeling needs of a particular domain would then be covered by a composition of multiple constituent models. Multiple domains are then covered by different compositions of models. This concept is what we refer to as model federation.

A model federation avoids multiple independently specified versions of the generic vocabulary and semantics. But it is still allowing specifying domain and use case specific additions. It enables an evolution towards more advanced features through new models that are added for replacing older versions in the federation. It also allows expanding into further domains by offering additional models that primarily specify the domain specifics. Inter-domain compatibility is reached by using the same common generic model and derive the extension models from it.

The root model in intent modelling that defines the generics domain and use case independent concepts, vocabulary and semantics is referred to as "intent common model". IG1253A proposes an intent common model intended to fill this role in the federation of models. A model that defines additional modelling features by extending and specializing the intent common model is referred to as "intent extension model". IG1253B defines a set of intent extension models. In addition, the model federation will typically contain models such as RDF, RDFS and OWL that define the general modelling base the intent related modes are built on. Models, such as XML schema define generic data types and the OWL time ontology might be used to gain expressiveness about time concepts. This means the model federation can contain additional general purpose models not specifically designed for use in intent.

## 14.1. Models within an intent model federation

Conceptually all intent in currently discussed use cases are very similar in their general structure and to a great extent also regarding needed expressiveness. For example, many intents need to specify something that needs to be delivered. In some cases, this something can be a service, or a product as ordered by a customer. In other cases, it can be a network slice or a network function. In any case the intent needs to describe all properties of the thing that needs to be delivered. It depends on the application domain and the corresponding intent handling scope, what exactly the thing is and which properties best describe it, but it is always very similar expressiveness with respect to needed vocabulary and its semantic mapping. Often the intent can simply refer to an entry within a catalog system.

Another example would be the definition of non-functional requirements. Often this is done by setting a target value or value range based on KPI. For example, intent related to services might require that service availability does not fall below 99.9%. An intent targeting a network slice might require a minimum available throughput of 1 mbps per user session. While the KPIs being used can differ significantly between domains, non-functional requirements are typically expressed by a numerical value that must be reached or exceeded to be compliant.

The conclusion is that many use cases of intent modeling only require generic modeling features combined with domain specific details. In this context it is a clear goal to avoid unnecessary divergence in intent standards and keep intent-driven operation compatible across domains. This means intent expression should be based on generic modeling features as much as possible and only introduce special features if the needed meaning cannot be achieved otherwise. Intent modeling can naturally not be kept domain independent, but it is still possible to distinguish common and domain specific aspects and manage the modelling features separately.

The intent common models defined in IG1253A is the proposal by the Autonomous Networks Project at TM Form for a common generic intent model that can act as generic root model for intent and intent reporting. It is designed to be the base of all extensions and specializations in intent modeling. The intent common model defines common modeling artifacts. It specifies for example the classes of intent and intent report. All intents are objects of the intent class. Furthermore, the intent common model defines the expectation class and related common properties. Expectation objects are distinct requirement expressions. For example, a delivery expectation object allows to specify a requirement to deliver something, for example a service, a function or a slice. A property expectation can be used to specify a requirement by setting a metric or KPI based target. These are examples of sub-classes of the expectation class.

Intent extension models can, for example, define further expectation sub-classes to introduce new types of requirements that cannot be expressed with the generic semantics covered by the intent common model. We refer to a model as intent extension model if it is based on the intent common model and extends expressiveness by relating the new modeling artifacts it introduces to the common artifacts defined in the intent common model. Defining a new sub-class of expectation is doing exactly this. It creates a new type of expectation derived from a generic expectation class of the intent common model.

Intent information models are another category of models within a federation. They add vocabulary but without necessarily being specifically designed for use in intents. For example, a KPI information model defines a set of KPIs and the identifier the KPI is referred to. This model might be designed as a general purpose collection of metrics definitions as part of the design specification of measurements and analytics systems. Intent specific models including the intent common model therefore not a base model for it. Nevertheless, the defined artifacts, for example a KPI, are used in intent expression, for example to set a KPI based goal within an expectation statement. For example, the property expectation class from the intent common model would be combined with a KPI defined in a domain specific information model to create a domain-specific requirement. This means that already existing information models, which were created independently of intent-driven operation might be directly useable in intent expressions and therefore become part of the model federation.

Intent objects are not explicitly typed. The use of certain expectations and the use of domain and use-case specific information models imply a certain purpose that corresponds to a type of intent. However, there is no need to explicitly make this type a part of the intent model. Typing of intent stays therefore a useful concept for documentation and human understanding but does not add semantics to the intent object. Therefore, the classes of intent and intent report do not have sub-classes.

## 14.2. Governance and management of model federation

We propose to apply the concept of model federation to intent modeling across the telecommunication industry and create a unified way to express and operate with intent. Ideally this is applied and adopted by all SDOs and projects working on domain and use case specific intent expressiveness.

The goal of the detailed proposals for intent modeling according to the Autonomous Networks Project is to not limit the authority of domain specific standardization work groups in other SDOs. The governance is kept to a minimum. In fact, no further coordination is needed as long as a common generic intent model is used as base and domain specific models' extent it. The authority over the details of the extension models is entirely with the SDO and project that has defined it.

With respect to the model federation in various domains, the intent common model described in IG1253A is the only model that is mandatory in the federation. It is defining purely domain and use-case independent expressiveness. The use of the intent common model as root of intent modeling makes the resulting intent compliant with the concepts of intent-driven operation defined by the Autonomous Network Project.

The design and evolution of the intent common model would ideally be a cross SDO effort and this can be driven by member companies who contribute to multiple SDOs as well as formal agreements and statements of intent between the SDO organizations.

It should become good practice to separate expressiveness that is specific to the operational domain from expressiveness that has broader applicability. The latter should be modeled into the intent common models, or as optional feature into its own extension model. A fine-grained partitioning of modelling features into multiple distinct models can avoid parallel developments and is common practice in the RDF modeling community.

An SDO with the mission to define the operation for a particular domain would not only create the respective intent extension models. Usually, this domain will contain one or multiple instances of intent management functions. The SDO would define the responsibility scope of the intent managers within this domain and define the models it is recommended or required to support. This means the definition of a standardized intent manager includes the specification of its model federation. It should define the minimal set of mandatory models and it can recommend further models and alternatives.

It is explicitly not necessary to publish intent extension models or proposed sets of models a domain specific intent manager needs to support through TM Forum or seek review or approval for these extensions from TM forum. Every SDO can use its own publication channels and processes and has full authority about their contributions.

An intent manager capability profile as introduced in IG1253D allows to describe the model federation a particular intent manager has implemented support for. Using the services of the intent manager registry, an intent manager can publish its profile.

## 14.3. Guidelines for intent extension models

Model federation for intent modeling as proposed by TM Forum Autonomous Networks introduces a lean governance and minimal central coordination of the models proposed to be used for expressing intent. We propose a common intent common model that defines the common and domain specific aspects of intent. Any SDO and workgroup can decide on its own to define intent extension models and information models to be used for intent-driven operation in that domain. There is no strict need for cross SDO approval or central coordination as long as the extensions to intent modeling are based on the common definitions and principles as defined in IG1253 and as long as the intent common model defined in IG1253A is considered a common foundation. The extensions must be done in a compatible way.

The requirements for creating intent modeling extensions with preserved compatibility are:

- Extension models are built against the intent common model. This is the common domain independent foundation of intent modeling. This can be done using the classes and properties already defined in the intent common model and referring to them in the definition of new artifacts. An example for this would be the definition of new sub-classes of expectations.
- Modeling artifacts that are potentially useful for multiple domains should be contributed into the intent common model or put into generic intent extension models rather than being kept in the domain specific intent extension model. This is not a requirement, but highly encouraged in order to avoid introducing multiple redundant artifacts, which all address the same concern. This would be the part of the proposed model federation where good collaboration between SDOs is encouraged.
- Extension models do not introduce artifacts that contradicts the definition of intent, or the principles of intent-driven operation. For example, artifacts that allow imperative statements within intent would violate the definition that intent is purely a declarative expression of requirements, goals, and constraints without specifying how to achieve them. So, a general agreement on definition and concepts would be preferential.

## 14.4. Model federation examples

### 14.4.1. Mutual agreement on models between intent managers

Federation of models allows intent managers to implement a distinct set of models to be used in the intent and intent reports it sends and receives. This means two intent managers that want to exchange intent and intent reports need to agree on the models used. They both can only use modeling features in their communication that both implement support for.

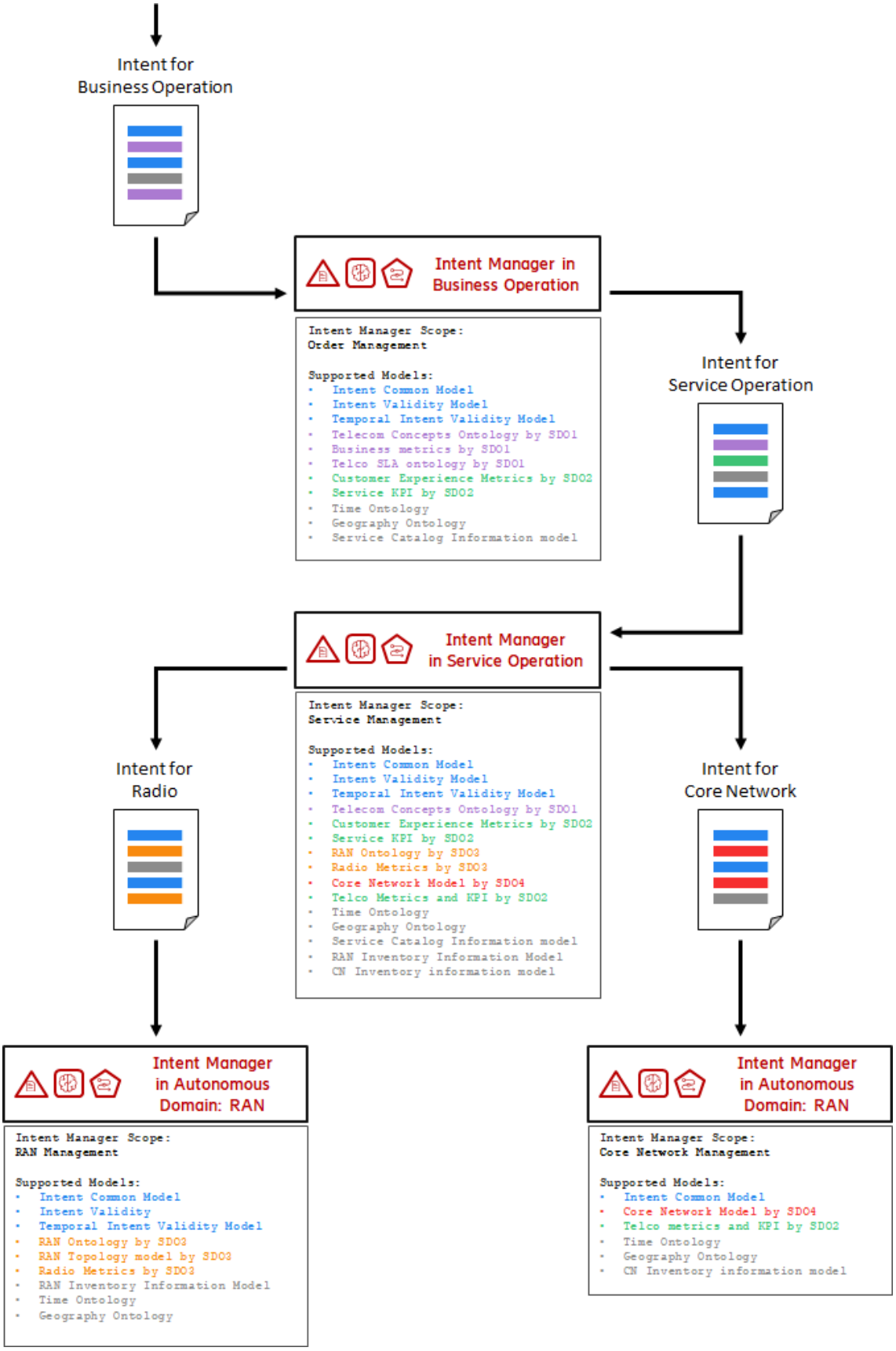


**Figure 14-1: Example of different domain specific model federations**

The example in Figure 14-1 shows four distinct intent managers with a specification of their scope and the related set of models they support. All of them support the intent common model in combination with a set of other models. For example, the intent manager responsible for service management would receive and handle intent coming form an intent manager allocated in business operation and responsible for order management. This intent can contain requirements about service KPI and customer experience metrics, because both intent managers understand the same models regarding these topics.

The example intent manager on the service operations layer uses subsequent intent to put requirements on RAN and Core Network Operation. This means that the service level intent manager needs to implement models that allow the formulation of RAN and Core Network requirements.

In this example the business layer intent manager is not concerned directly with details of RAN and Core Network operation. The service layer provides in this respect an abstraction of the resource operation.

Furthermore, in this example, RAN and Core Network are distinct domains that do not exchange intent with each other. This means, it is not necessary that they implement and support each other's domain specific models. Please note that this is an example and it is explicitly allowed to have horizontal use of intent within the same system layer. From intent modeling perspective the only prerequisite is that both involved intent managers have respective overlaps in the set of supported federated models.

### 14.4.2. Practical expression of model federation within an intent notation

The following example shows the use of constituent models within an intent. The color coding distinguishes constituent models and the vocabulary they contribute within the model federation.

```
@prefix icm:  <https://tmforum.org/2020/07/intent/> .@prefix
tel:  <http://sdo1.org/TelecomConcepts/> .@prefix
met:  <http://sdo1.org/metrics/version2/> .@prefix
sli:  <http://sdo2.org/2021/03/SliceIntent/> .@prefix
slk:  <http://sdo2.org/2019/SliceKPI/> .@prefix
slm:  <http://sdo3.org/v1.1/SliceManagmet/> .@prefix
cat:  <http://operator.com/Catalog/> .@prefix
ope:  <http://operator.com/Inventory/> .

ope:ExampleIntent2021031100002   a icm:Intent ;   icm:hasExpectation ope:E1,
ope:E2, ope:E3, ope:E4 .
 ope:E1    a icm:DeliveryExpectation ;
    icm:target _:function ;
    icm:params ope:P1 .ope:P1   a icm:DeliveryParam ;   icm:targetDescription
cat:amf .ope:E2   a icm:DeliveryExpectation ;
    icm:target _:slice ;
    icm:params ope:P2 .ope:P2   a icm:DeliveryParam ;   icm:targetDescription
cat:SliceTypeA .ope:E3   a icm:PropertyExpectation ;
    icm:target _:function ;
    icm:params ope:P3 .ope:P3   a icm:PropertyParam ;   tel:subscribers [
icm:upTo 1000 ] ;   met:availability [ icm:atLeast 99.9 ] .ope:E4   a
icm:PropertyExpectation ;
    icm:target _:slice ;
    icm:params ope:P4 .ope:P4   a icm:PropertyParam ;   slk:latency [ a
slk:Latency ;                 icm:atMost 10 ] ;
    slm:sliceState [ a slm:State                    icm:oneOf slm:up,
slm:available ] .ope:E5   a sli:LinkExpectation ;   icm:target _:slice ;
    icm:params ope:P5 .ope:P4   a sli:LinkParameter
;   sli:connectingEndPointOf _:function .
```

In this example the intent object is built using multiple models. Each model is represented by a unique URI through which all vocabulary defined in the model can be referred to.

At the beginning of the intent object each used model is specified and a prefix is assigned. This is common practice in TURTLE notation of RDF/RDFS/OWL based models. It makes the notation readable by shortening the long and repetitive part of the full URI. For example, metrics from version 2 of the metrics model defined by sdo1 would be identified by a combination of the metrics name as used in the model and a prefix that identifies the model.

The example intent object is defined internally within the domain of an operator and therefore has a URI based on the operator's domain.

All objects used in the intent expression with the prefix "icm:" are from the intent common model as defined in IG1253A. The intent common model is the root model for all intent definitions. It defines base classes for intent objects and its internal structure. It defines for example the class "icm:Expectation". An expectation is a distinct type of requirement. An intent is then a set of multiple types/classes of expectation. The intent common model defines expectation classes that are generic and not domain or use case specific. This means, what they allow expressing is useful in many domains and constitutes a general pattern. For example, the expectation sub-class "icm:PropertyExpectation" allows to establish a requirement based on metrics including KPI.

Using KPI for expressing a quantitative target is a pattern that is frequently used in many domains. The actual metrics and KPI used are domain specific. Other models are then used to extend and complement the expressiveness with domain specific extensions. For example, a standard about telecommunication metrics would introduce a KPI and define it in detail. The intent model can now link to the respective standard and point to the KPI needed. For example, the slice latency referred to by "slk:latency" points to the latency metric from the slice KPI model as defined by sdo2. The requirement for the slice is therefore expressed using the generic property expectation in combination with the domain specific KPI.

Another possible extension would be the introduction of further classes of expectation. In the example above, the "sli:LinkExpectation" is a sub-class of "icm:Expectation" as defined in the slice intent model by SDO2. Here SDO2 is considered to be responsible to standardize slice management. Based on the intent common model a work group within SDO2 has created a domain specific extension to provide the expressiveness needed for intents concerning slices.

## 14.5. Linking to and from other modeling standards

In telecommunication and IT implementations UML and Yang and other standards are frequently used for describing interface information models and data schema. This means the intent manager exists in a heterogeneous environment of many modeling standards. Intent management needs to consider all aspects of the system state that might be relevant for fulfilling the intent. For example, a metrics based requirement can only be managed if this metric is measured. A functional requirement is typically formulated using pre-defined functions that reside in a catalog. Inventories keep track of artifacts such as deployed software instances, used resources and topology information about the managed network. All this data and information is relevant for the tasks of an intent management function and intent might express requirements with

and about it. Therefore, Intents often require referring to objects and artifacts described by various models based on other modeling technologies than the intent model.

### 14.5.1. Referencing with constructed IRI/URI

Referencing to an artifact from intent models is based on IRI and URI. Referencing to models based on other technologies than RDF bears the main challenge of creating a respective IRI. The main issue is that non RDF models often miss a globally unique identifier and have local references instead.

Distinguished names (DN) are a frequently used format for unique identifiers of objects. It is standardized by ITU [x500] and for example the base of referencing in LDAP and used in 3GPP [32.300]. Distinguished names in general only provide identification in a local context and namespace. There are however proposals for mapping between distinguished names and IRI/URI. For example, 3GPP describes a mapping scheme in TS 32.158 [32.158].

URIs are globally unique. For this reason, only a globally unique DN with a domain component can be mapped directly into a URI. The mapping rules are as follows [32.158]:

- The DN prefix is mapped semantically to the authority component of the URI. The syntax of the DN prefix is modified to match the syntax of the authority component.
- The Local Distinguished Name (LDN) is mapped semantically to the path component of the URI. The syntax of the LDN is modified to match the syntax of the path component.

However, when DNs do not contain a domain component a mapping scheme is still possible by assigning a proxy domain component to the local context by convention.

For example, the objects in a catalog are identified through Local Distinguished names that do not contain domain information. The inventory is however deployed within an operator's network and therefore the operator's domain can be used to narrow down the context. Then there might be multiple instances of the inventory within the network. This can be represented through an initial path component in the URI string. This way the individual instance of the inventory could be uniquely identified. Finally, the LDN can be appended to the partial URI similarly as described in [32.158].

This URI creation scheme works if the owner of the domain (e.g., the operator) consistently assigns and manages the naming for identifying local nodes and instances of functions that exist within its domain. There are potentially different algorithms that can be used to formulate the URI and it is a choice of the domain owner, which one to use. In any case the resulting URI would be globally unique as it contains the unique identifier of the domain, and a path managed by the domain authority. The result is globally unique and may therefore be exposed outside the domain, if necessary. For example, if the operator and their customer exchange intent and intent reports that were modeled according to IG1253 and its sub-documents. The system that needs a URI for unique referencing would then be able to assemble the URI itself from the local naming and addressing conventions.

The IRI/URI would be composed of:

1. The administrative domain of the system in which the addressed object resides. For example, the operator's domain in the form of "https://www.operator.com/"
2. An address of the targeted system within that domain. For example, "/NorthernChina/ServiceCatalog/"

3. The local identifier (LDN) of the addressed object within the system that manages it. This would typically follow the local information model and addressing scheme. For example, "/Services/ManufacturingServices/ExampleService0001". This refers to the individual service "ExampleService0001", which resides in the "ManufacturingServices" table within the "Services" section of the catalog.

All together, these parts form a URI, which addresses the Example Service defined in the Northern China catalog of the operator with the domain "operator.com"

```
"https://www.operator.com/NorthernChina/ServiceCatalog/Services/ManufacturingServices/ExampleService0001"
```

This URI can be directly used in RDF based models and in the knowledge base of the intent manager to reference this object. It is however only a proxy URI and accessing this object requires an adapter that translates this URI into a query specific to the interface exposed by this catalog implementation. Building generic adapters is possible. Using the naming convention set by the operator it is then possible to use the URI and route requests for the identified object to the right local node. DNS can for example translate the URI into the IP address of the node by which the addressed object is managed. Within the context of that node the LDN component carried by the URI would uniquely address the object.

The method of creating artificial proxy URIs might not always be practical or expressive enough when dealing with dynamically changing data and resources. A more capable alternative is the use of selectors proposed by W3C in [select].

Objects to refer to are often represented and by data and organized in databases. W3C has proposed how to map data allocated in relational databases in [rdb2rdf].

## 14.6. Model federation as cross industry use case enabler

The intent common-model is also designed to allow extensions into other industry and technology domains than just telecommunication. For example, domains such as IoT or automated manufacturing can certainly also profit from the paradigm of intent based operation. The details are out of scope of this document, but the proposals for intent modeling and in particular the model extension and federation mechanisms would allow exploring and define intent in a broader application scope. This means intent as defined here has the potential to become a universal base for autonomous operation across classical industry borders. This would allow automation of truly cross industry use cases where intent can have a central role in communicating and distributing requirements as well as setting up cross-domain control loops. This helps to manage for example the diversity in IoT and utility applications. Consequently, standardization organizations and work groups from any industry with respective needs for intent-driven operation would be invited to participate in the proposed federated modeling of intent.

# 15.Overview of specified models

The following models are defined in the current release of the IG1253x family of documents:

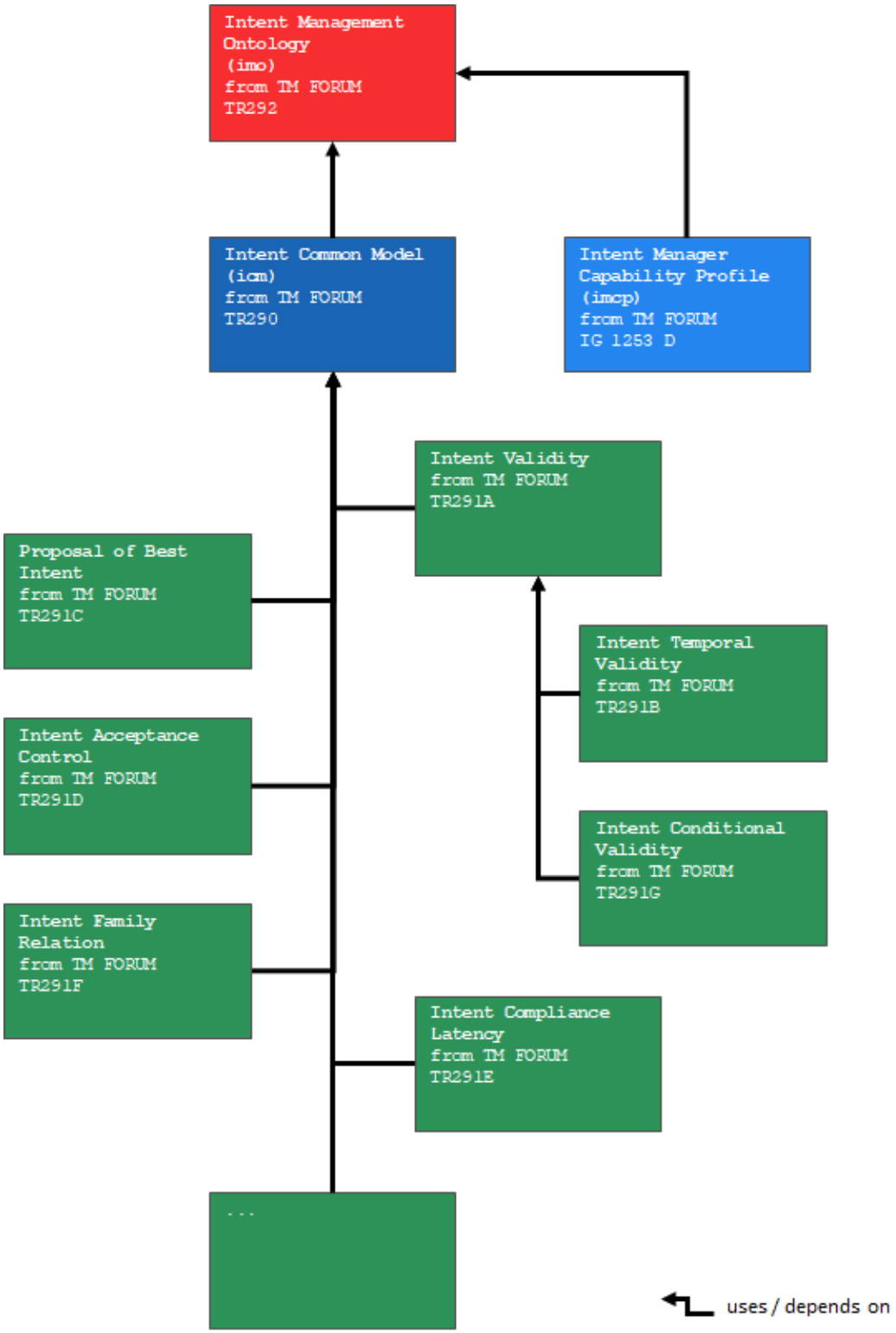


**Figure 15-1: Models specified in IG1253x and their dependencies**

The intent management ontology provide general vocabulary about intent management.

The intent common model is the root for expressing intent and intent reports. For a particular intent manager, it is typically federated with a set of intent extension models that define optional and domain specific additional vocabulary and semantics.

The intent manager capability profile model defines the details of how an intent manager can specify its capabilities and scope for sharing this information with other intent managers through an intent manager registry service.

All these models are based on the RDF/RDFS modeling family. OWL was not used yet, but it might be in future versions.

In addition, further models are used for specific needs. For example, the intent common model, the intent temporal validity and intent compliance latency models need to express points in time or time durations. They gain the respective vocabulary from the W3C time ontology. Basic data types, such as strings and integers are provided by XML Schema.

# 16.Intent related closed loops

Intent managers are involved in three different types of control loops shown in Figure 16.1.

**Intent control loop**
This is the control loop related to the intent life cycle and executed between two intent managers typically in distinct roles of intent owner and intent handler. This control loop is executed over the intent interface. It consists of two basic actions: Intent is defined and communicated in one direction to set requirements. Intent reports close the loop by communicating the fulfillment and handling state back to the origin of the intent.

**Intent manager inner loop**
Intent managers have to continuously check that their goals are fulfilled and take actions if not. This is a typical assurance control loop. The subject to be assured are mainly the requirements, goals and constraints defined by intent. This means that the Intent manager inner loop interacts with the intent control loop/

**Other control loop**
Intent managers exist in an environment that is not fully adapted to intent based operation. This means, there are many control loops that are based on other interfaces than the intent interface. The intent manager can participate in these loops if this is needed for assuring that the intent expectations are met. This means and intent manager can act through action on other control loops as part of its handler role. An intent manager in the owner role acts towards other intent managers through the intent interface by defining and sending intent. By doing this it is initiating intent control loops to control the fulfillment of its instrumental goals. The terminal goals of an intent manager might however come from the intents it is handling, but they can also originate from other interfaces as part of other control loops.

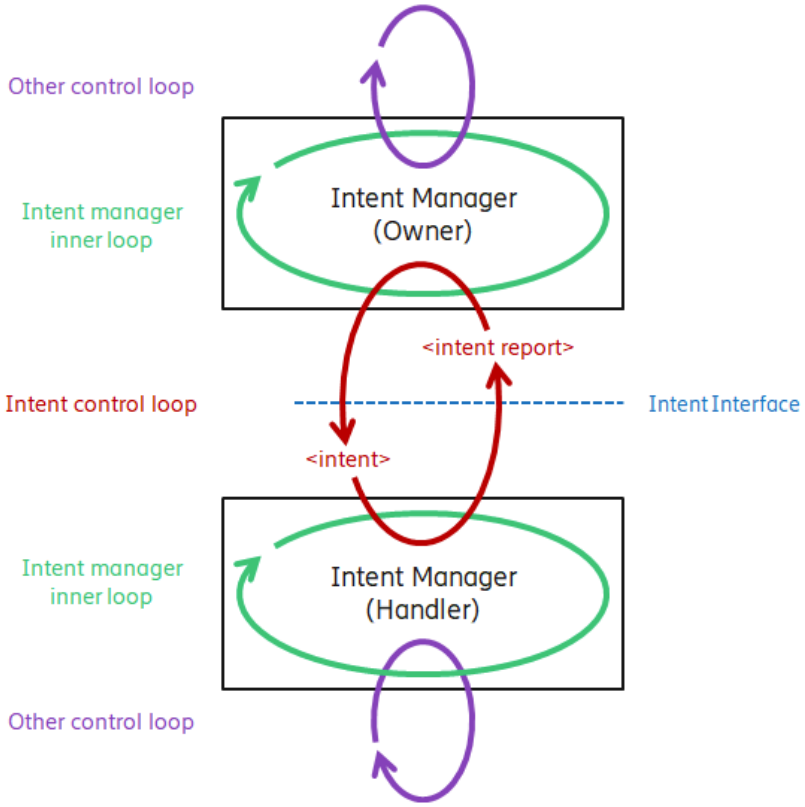


**Figure 16-1: Control loops related to intent managers**

The inner control loop of an intent manager in the role of intent owner is mainly concerned with control of the intent life-cycle. This means the intent owner uses intent reports coming from the handlers to monitor if the intent fulfillment is meeting its requirements. It can act for example by adding or changing intent.

The inner control loop of an intent handler is concerned with taking action to assure that the requirements from the intent are met and the result is reported. In the handler's inner loop, it continuously analyses the state of the underlying infrastructure and decides on action plans and strategies to improve its intent fulfillment state. Actuation of the handler would be through further control loops. They can be intent-driven or the handler decides to act by sending further intent, thus assuming an intent owner role for this intent. But action through participation in any other control loop is equally possible. In this respect there is only one control loop within an intent manager covering all handled and owned intent, because all requirements have to be met and potential contradictions and conflicts between them need to be detected and resolved, for example, through prioritization.

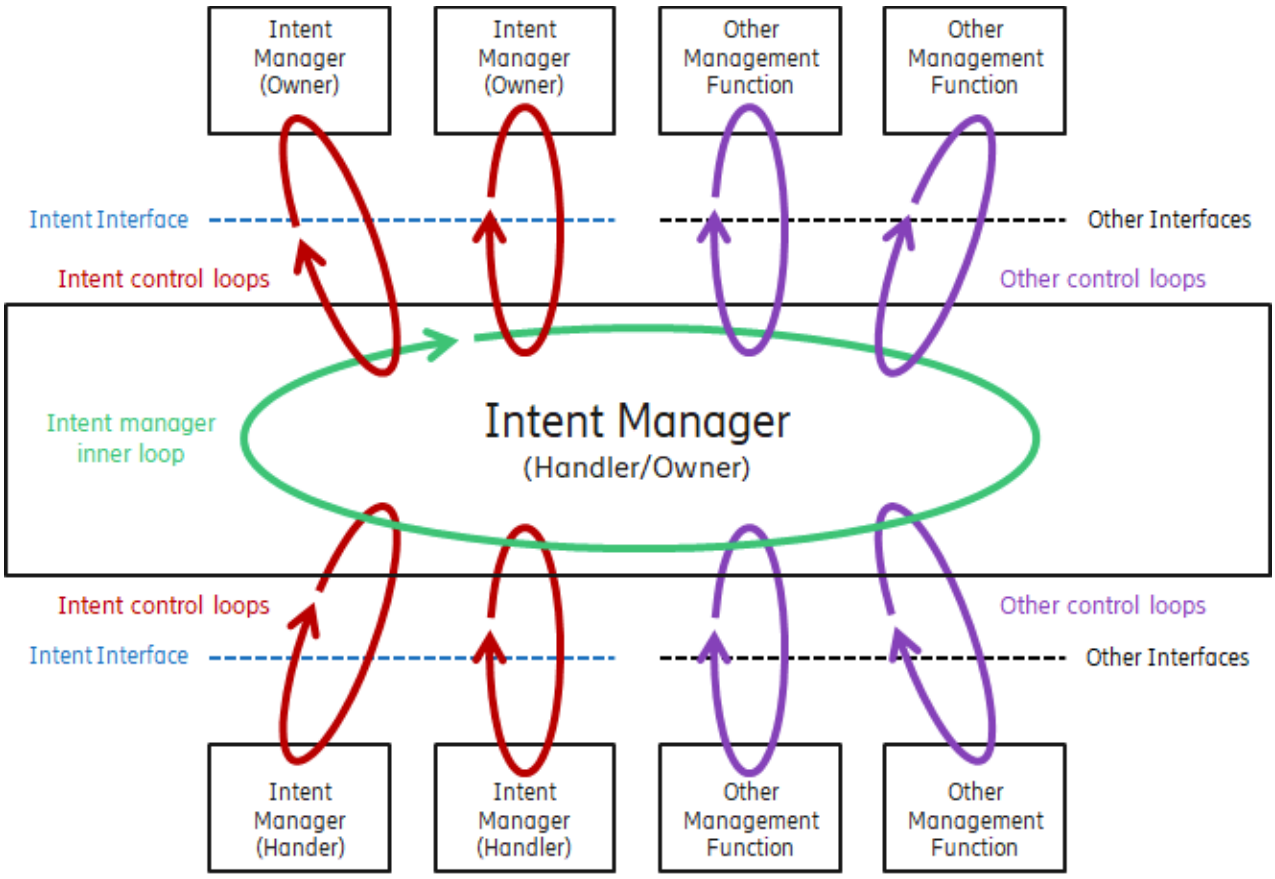


**Figure 16-2: Control loops within and around an intent manager**

An intent manager typically participates in multiple intent control loops assuming the role of intent handler for some and intent owner for others. In addition, it participates in any number of other, non-intent based control loops. This situation is illustrated in Figure 16-2. If an intent manager has the dual role of being handler of some intent and owner of others, there is still a single inner loop. It is part of the concerns of the handler role to act according to the received requirements and these actions constitute an owner role if they involve the creation and sending of further intent. The concerns and therefore the loops converge into a single one.

## 16.1. Interaction with real-time control loops

Intent handling typically involves many distinct activities within the handling control loop. For example, the system state needs to be collected through measurement and analytics. Furthermore, solution proposals and action plans would need to be generated, evaluated, prioritized and acted on. Also, formal obligations of the intent mechanism, such as the generation of intent reports would be executed. Many of the tasks in intent handling involve evaluation capabilities or even predictive models and assessment of alternatives and risk or actions. Complex processes like this can usually not be executed, while also meeting challenging real time requirements. System reaction times would stay on the level of best effort. If intent handling requires very low latency reaction times or if challenging real time requirements apply, the implementation of intent handling might follow a different approach. Low latency control loop require simpler models that directly translate an observation into actions with as small intermediate steps as possible or controlled by managing time budgets and actuation deadlines. Extensive evaluations and exploration of alternatives are usually not possible if the real-time and action latency requirements are demanding.

Low latency requirements can be met if the intent handler interacts with one or several low latency real-time enabled control loops. The intent handler can act on intent by putting specialized control loops in place. These control loops are designed to meet the real-time requirements by deciding and acting in a distinct closed loop without involving the intent handler in every decision and action. Intent handlers are not in the low latency loop and can therefore also not apply prioritization and action approval to each action directly. The intent handler needs to trust the implementation of the control loop to act in a preferential way. What the intent handler will do however, is to monitor the action results and check if they were preferential for the composition of intent present in the system. It can retrospectively assess if the actions that were performed were the right and optimal ones. The intent handler can then act and influence future actions of the autonomous real-time control loop by changing its configuration or by replacing the control function that is in charge of the real time control with a more capable implementation or by updating the models it is based on. This means the intent handler would be able to indirectly assure and improve the real-time control.

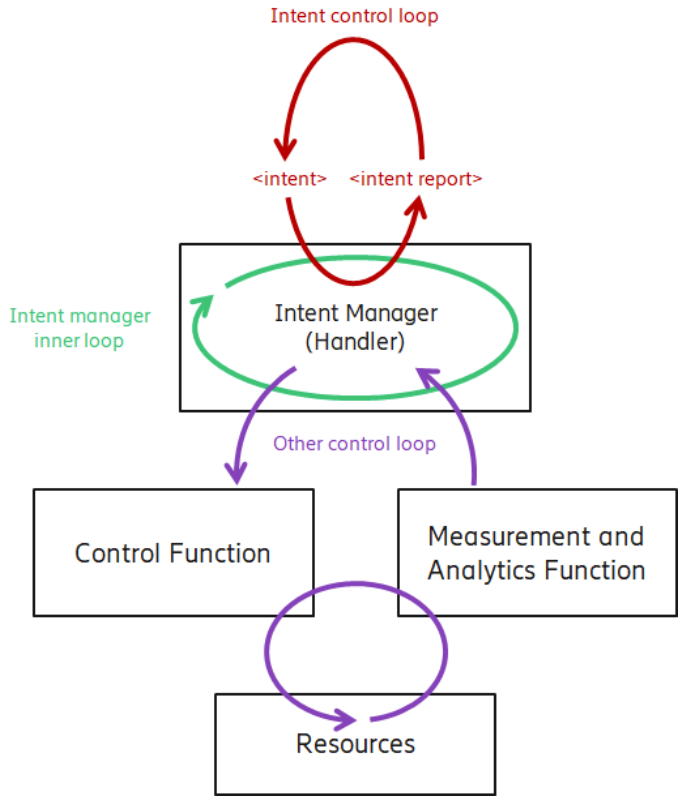


**Figure 16-3: Interaction with other functions that themselves participate in control loops**

Figure 16-3 shows a typical scenario of an intent manager participating in a non-intent based control loop and acting through other than intent interface. Here the intent manager in the handler role is acting by configuring or ordering action from another control function within the domain. This control function is then controlling a set of resources. Their performance and state is observed through measurement and analytics functions. The intent manger also obtains its information about the system state through an established measurement and analytics infrastructure. This closes the loop from the perspective of the intent handler.

This setup allows an intent manager to indirectly participate in real-time or near real-time control, while itself does not have an implementation able to meet real-time operation requirements. The real time control is happening between the other control function and resources, while the intent manager observes the progress and can interfere and influence by replacing the real-time enabled control function or changing its configuration and models.

From the perspective of real-time control functions, this means that their implementation does not directly need to consider intent or include an intent management function. It can stay a specialized control function with a highly efficient implementation. It is typically doing a control task with narrow scope, but this allows it to do it well and efficiently. The generalization and allocation of its tasks within a broader environment of diverse requirements can be left to the non-real time intent management system.

# 17. Intent from natural and domain specific languages

Intent specified and managed according to IG1253 is not the only way to express and manage intent. There are many use cases where the use of other language for intent expression and methods for managing would be a better choice. For example, on the human-machine interface presenting and expressing intent using natural language or a domain specific language (DSL) made for this purpose can lead to a more immersive and intuitive experience for human users. This is particularly the case for users, who are not technically trained and not familiar with formal notations of models. These users should not be exposed directly to formal intent modeling. Other human users in the role of technician, who oversee the autonomous systems for the operator might however prefer formal notation and require the clarity and precision it provides. The point is that humans are very diverse in their abilities, preferences and needs and this can be reflected with a range of languages and methods used.

In the context of this discussion, we refer to intent expressed according to IG1253 as "formal intent" and intent expressed using other domain specific languages including natural language as "DSL intent".

The details of how DSL intents are encoded and managed are best defined by the party that specifies interfaces and interactions within a domain. This is typically SDOs and work group specialized in the respective domains. Therefore, we consider the details of expressing and handling of DSL intent to be out of scope of the IG1253 set of specification documents. Nevertheless, this document acknowledges that interwork with DSL intent is required and its integration into autonomous network operation needs to be addressed.

This chapter proposes modeling of the touch-points between the use of formal intent and DSL intent. It is doing so without implying specific details in the modeling and expressiveness of DSL intent as well as the management functions and procedures involved.

The main idea is that DSL intent is used only on the periphery of the autonomous network. Inside the autonomous network intent is always expressed as formal intent according to IG1253 and this intent exclusively used between intent handlers and owners. Formal intent is always the reference for all intent handling. DSL intent is solely used for compatibility with external systems and interaction with human users, but never propagated directly to intent handlers. This is possible, because formal intent is defined in IG1253 with a model federation approach allowing it to cover the needed vocabulary and semantics of multiple domains.

DSL intent can be used by a human user or a technical system.  This would be a system designed and implemented without considering formal intent according to IG1253. A human-machine interface is typically realized through a frontend implementation for direct exposure of intent to human users. It offers presentation and editing capabilities according to the needs and abilities of the targeted human audience. While formal intent can be used directly also on this interface as there are standard formats for its direct editing and presentation, but other DSL for intent expression might be preferential depending on the targeted human audience. This interface is also not necessarily based on textual representation but can include other media such as audio and speech.

Other technical systems using DSL intent might be designed according to a broad range of other standards or system vendors' own specifications. Typically, the intent models used in these systems only cover the expressiveness as needed for domain the system is designed to operate in. An example for using domain specific intent would be an ordering system belonging to the customer using DSL intent to communicate the customer's needs to the operator's autonomous network.

DSL intent is used to express and convey requirements, goals, and constraints.
We consider that DSL intent complies with basic principles of intent-driven operation. It only contains declarative requirements and goals according to the intent definition of IG1253. This particularly means that it is free of imperative artifacts that prescribe actions or particular solutions.

DSL intent is used as part of a control loop
This means there is in general the need to communicate back to the entity the DSL originates from and provide feedback about the intent handling progress and status.

DSL intent is not necessarily free of ambiguity.
Ambiguity means that the semantics expressed by intent are not necessarily interpreted the same way by different systems. Usually this can occur if not all content of the intent is completely covered by standardized formal models. Natural language is a good example where a statement can be interpreted differently depending on context, situation and parties involved in the conversation. Correct understanding of a statement implies that originator and recipient have the same contextual knowledge and based on that agree on the statement's meaning. A human statement in natural language therefore assumes that the receiving system has respective models that match the context and meaning implied by the human and cover all its aspects. This typically requires that the system is able to distinguish who has made the statement and in which context. Misinterpretations of the statement are hard to avoid if any aspect relevant to the meaning gets lost between systems or is not covered by models and implementation. This also means that two systems receiving the same statement might not necessarily agree on its meaning. Only standardized and complete models would avoid this ambiguity.

Formal intent according to IG1253 is ambiguity-free, because all its expressiveness is covered by formal modeling. This means two systems sharing intent according to IG1253 will arrive at the same interpretation of its meaning. When dealing with potentially ambiguous DSL intent, this means that a single point of interpretation is preferential. This interpretation and its representation as formal intent would then become the reference for all subsequent decisions and actions in autonomous operation. The single point of interpretation should also be logically allocated as close to the originator as possible to capture all needed contextual information. Furthermore, a sequence of multiple conversion steps with multiple intermediate domain specific languages should be avoided because every step bears the risk of losing partial meaning. Even small losses per conversion can accumulate and entirely distort the message. Figure 11 shows the use of DSL intent and how it enters IG1253 based autonomous network operation through intent interpretation.

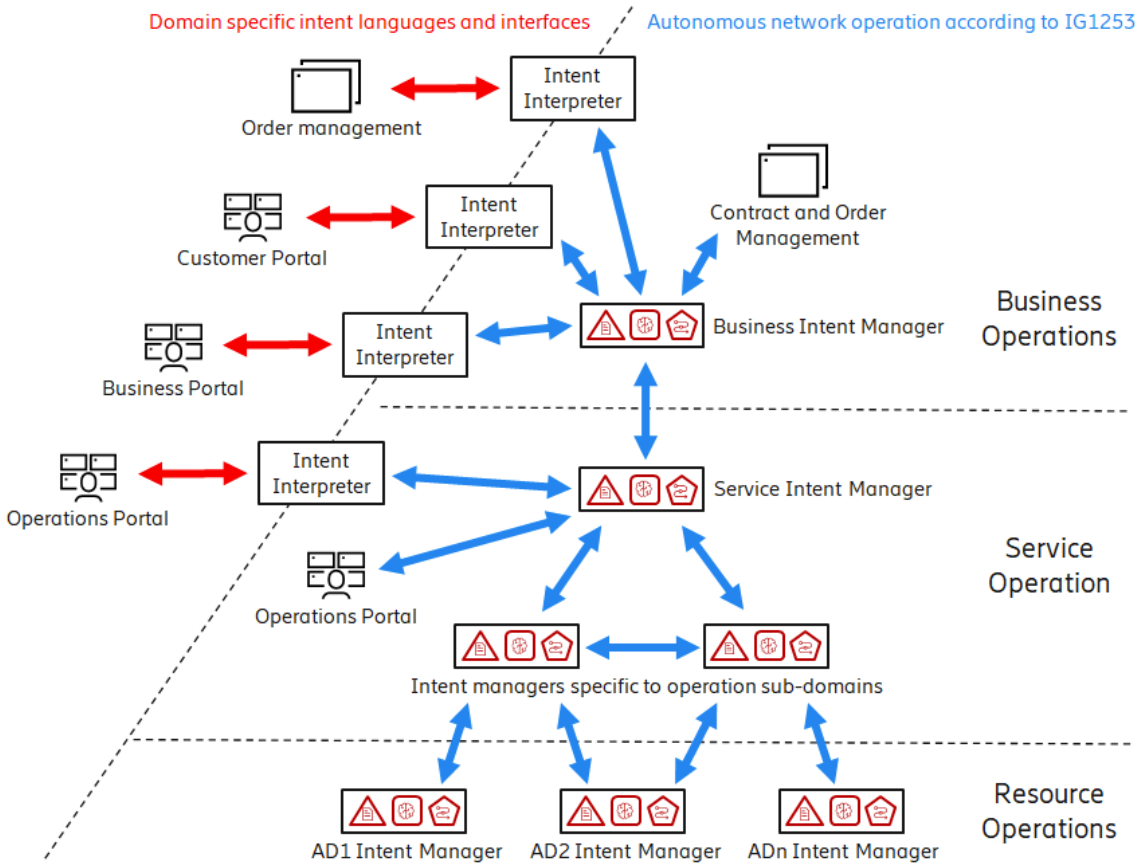


**Figure 17-1. Intent interpretation on the periphery of autonomous networks**

Also, on the interface that is using DSL and natural language there will most likely be a notion of owner and handler with distinct role in realizing intent management targeting intent-driven operation. It is not in scope of IG1253 to define these details. This is naturally left to the domain specific specifications. In this document we only go as far as defining the entry point of DSL intent through an interpretation point.

## 17.1. Modeling intent originating from domain specific languages

We propose to model the use of DSL intent through the generic management service "intent interpretation". There can be multiple different domain specific languages, and we would consider that each of them is modeled through a distinct variant of the intent interpretation management service. Here we only imply that this management service exists, but not the detailed procedures on its interface or other details of its implementation. In this respect the intent interpretation management service is placeholder for any service interface used to define and manage intent that is not the formal intent according to IG1253 and therefore not managed through the intent handling MnS.

The producer of the intent interpretation MnS is referred to as Intent Interpreter. Its main role is to receive the DSL intent and translate it into a formal intent according to IG1253. The corresponding formal intent would then propagate further into the IG1253 compliant autonomous network for handling. Formal intent rather than the original DSL intent would be used directly for operation and intent handling.

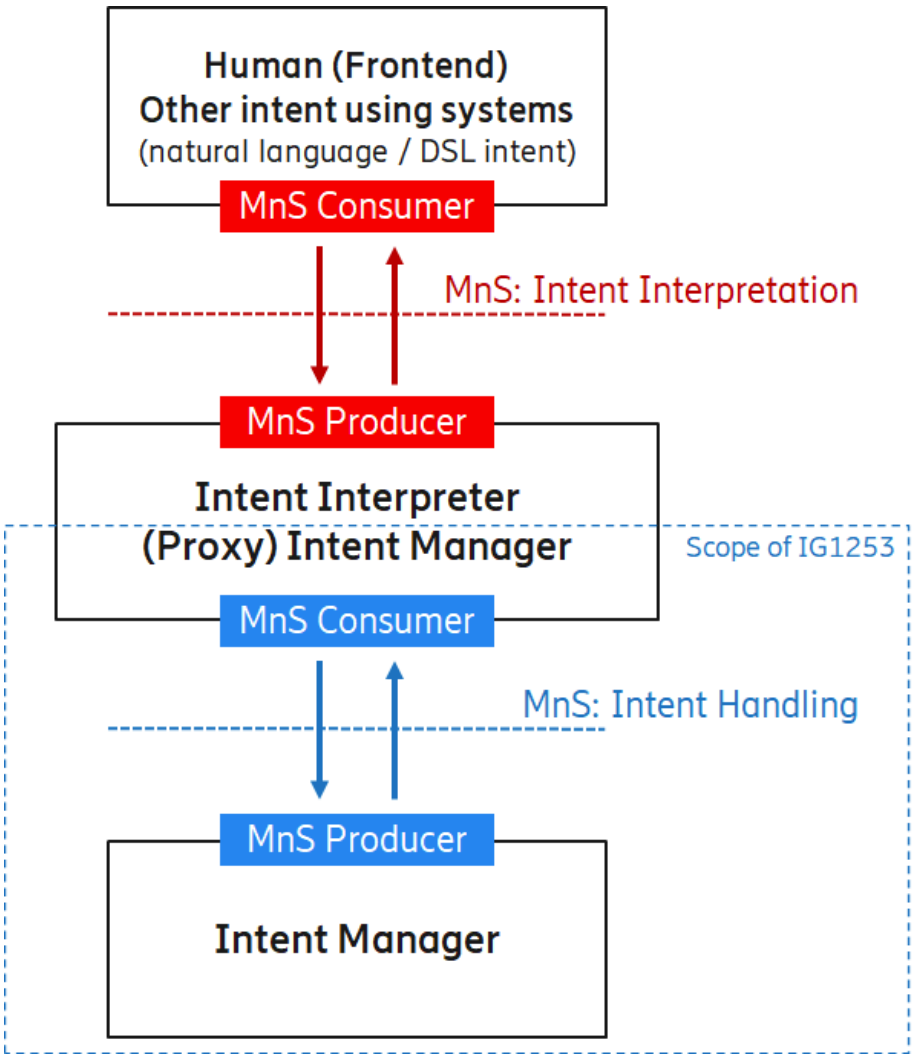


**Figure 17-2. Introduction of intent interpretation**

The intent interpreter produces formal intent from the DSL intent, and it needs to issue it into autonomous network operation. It would also need to receive intent reports and translate them back into the respective reporting formats and procedures specific to the DSL intent and its interface. This means that the intent interpreter becomes a consumer of the intent handling MnS. Consequently, it assumes the role of intent owner in the lifecycle management of the formal intent objects it has created from the DSL intent. It therefore becomes a proxy owner for the human or technical system the DSL intent actually originates from. However, the fact that it is a proxy owner and that the formal intent it provides originates from a DSL intent is entirely transparent to the intent handlers. From their perspective it is yet another owner, and they receive yet another intent.

The intent Interpreter is providing a suitable interface endpoint according to the needs and specifications of the DSL intent. This is captured by modeling it as a distinct management service. The detailed interaction procedures on this interface is defined externally and the implementation of the intent interpreter needs to provide a mediation between the interface procedures of DSL intent and the IG1253 intent interface.

Figure 12 shows how to model multiple sources of intent. Next to formal intent the picture also shows how to model natural language and DSL intent specified using another domain specific language.

Any DSL intent would be translated into formal intent through the intent interpreter. The intent interpreter assumes the role of intent owner for the formal intent. At this point all intent, no matter in which format it was originally provided would follow the specifications of IG1253. All intent would be lifecycle managed and handled through the intent handling MnS accordingly.

Figure 17.3 also shows the intent management function that handles intent from multiple sources and some of them did originally come from domain-specific intent through intent interpreter and proxy owner.

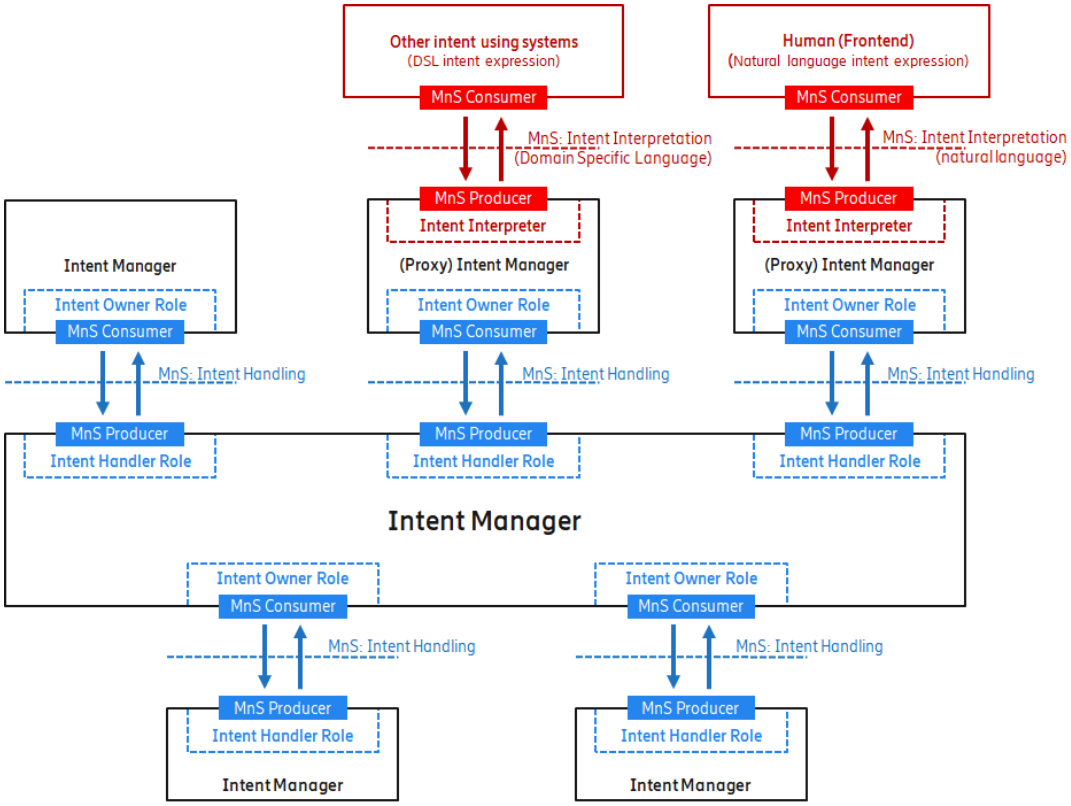


**Figure 17-3. Various sources of intent**

Intent provided through legacy interfaces can in principle also be modeled with the approach presented here. However, these interfaces often mix requirement and goals with imperative aspects such as policy triggers or process invocations. Interfaces that meet this criterion need a separate analysis with respect to their coexistence and interwork with intent-driven operation in Autonomous networks. This is out of scope for IG1253 for now but a topic that should be included at a later stage or investigated by another project.

# 18. Implementation aspects of intent management

## 18.1. Concerns addressed through intent versus implementation

Requirements and constraints can be specified by intent or handled implicitly by the implementation logic of the autonomous system.

The behavior of an autonomous system is primarily determined by the implementation of decision and action taking logic. This includes hard coded logic, but also more dynamically changing artifacts with more flexible life-cycles. Examples for that are policies, rules, or apps (e.g., rAPPs in ORAN). Also, machine learned models would fall into this category.

Implementation artifacts address implicitly many concerns and requirements already. In fact, these requirements are addressed at design time by a human system architect, developer or data scientist when implementing these artifacts. Deploying them into the autonomous system makes them available as solution capabilities.

Intent, by definition, does not include any imperative aspects that prescribe a particular solution. This means intent and any implementation logic that determines its handling are mutually exclusive.

Successful intent handling is always based on a good match between what an intent expresses and what the system implementation can do based on all its available solution components. Therefore, if intent cannot be handled successfully, the root cause might be that the intent handling does not have suitable solution artifacts and is therefore limited in its options. Please note that this is not a limitation of available resources, but a limitation in finding a solution for how to use these resources.

This first means that even if a concern is addressed through intent expressing a respective requirement, the autonomous system might still fail to act on it if it does not have suitable solution finding and planning artifacts available.

This also means, even if a concern and requirement is not subject to intent, the autonomous system might still consider it. In this scenario, all solutions the system finds would comply anyway.

For example, no intent might have expressed a requirement that strict tenant separation is applied in service deployments. However, all solutions options the system finds for service deployments nevertheless contain it. This requirement is implicitly addressed for example by policies involved in the solution finding process.

That a requirement is not communicated by intent is not necessarily a limitation. Not all concerns need to be dynamically changeable or allow being chosen by, for example, the customer or service provider. Other concerns and especially basic common sense can very well be addressed in the implementation of solution finding algorithms.

If, however an intent, which explicitly asks for sharing service instances between tenants, would be given to the same autonomous system described in the example above, this intent cannot be fulfilled. All available solutions do not consider this aspect as required. This first can mean that the operator explicitly wants to avoid the implied solutions. Not fulfilling this intent is therefore the right reaction. This requirement is wrongly given, and the autonomous system has protected the operator. This outcome would be reported back to the intent origin.

On the other hand, this new requirement might be a variant ask for by customers and the operator agrees. It reflects a new customization variant of the services sold to the customer. The rationale might be that this more resource-efficient deployment variant can be offered for a lower price. However, the operator's autonomous network is not yet able to handle this new variant. This causes a breach of autonomy, because in order to fulfill this new intent, the implementation of the system or some of its solution finding artifacts need modification. This is typically a task involving humans and requiring considerable lead-time to implement.

In this example, a system able to produce solutions for tenant separation as well as for service instance sharing would be more autonomous. It either has already more complex polices available that can produce both solution variants and select them accordingly, or it has the capability to analyze the new intent and therefore adapt its policies itself.

In any case this system does not assume anymore that the multi-tenancy concern is always addressed the same way. Consequently, it needs to be told by intent when either of the alternatives would apply. This shows how a concern can become subject to intent when the system expands towards more autonomous behavior. In this respect "more autonomous" is understood as more situations can be handled without human involvement.

## 18.2. Conflict detection and resolution

A system is usually given multiple intent from multiple sources. There are two levels of conflict:

**Explicit contradictions:**
Intents directly conflict with each other by expressing explicitly opposing or incompatible requirements. For example, one intent is stating that all network links shall be encrypted, while another intent is stating that all network links must not be encrypted. Consequently, at least one of these intents will get violated. Nevertheless, as long as these requirements are provided with context that resolves the conflict they can coexist. This can for example be a non-overlapping scope such as requiring encrypted links for one user group and no encryption for another user group. As long as the user groups do not overlap, there is no explicit contradiction.

Contradictions also disappear if one requirement includes another one. For example, if one intent requires at least 10 ms latency and another one for the same target at least 5 ms. There is no conflict because a solution exists that satisfies both. In this example, reaching 5 ms latency also satisfies the 10 ms latency requirement.

**Conflicting handling actions:**
In many cases conflicts are not obvious from the requirement itself but originate from conflicting actions being proposed in their handling. This will happen if the intents are realized with a common infrastructure and limited resources. For example, one intent might ask for increased RAN coverage and another intent for increased throughput delivered to users. Actions are proposed to satisfy these intents and both actions imply opposing reconfiguration of RAN cells. For example, by changing antenna tilt, but in opposite directions. Each proposed RAN configuration will improve the fulfillment of the two intents but degrade the other.

This indicates that intent handlers should be able to prioritize and therefore find the operational state that is globally preferential even if it means to degrade some intents. Please note that partially degraded intents would only occur if the handler does not have the means to satisfy all intent. All it can do is prioritize by maximizing global utility,

at least within its handling scope. In any case the partial degradation would be communicated to the intent owners to allow them to take action accordingly.

## 18.3. Intent expressing the wanted ideal system state

Intent is introduced as knowledge about the requirements and goals of a system. This can be understood as the wanted or targeted state the system should ideally be in. This means there is a range of preferred states that fulfill the intent, and it is the task of intent handling to bring the system into these states through suitable actions.

Intents are additive and therefore there are in general multiple intents the system is working to fulfill. This might not always be possible due to conflicts between the intents, the actions planned to fulfill them or the current availability of resources. This means that there are no states the system can reach that would fulfill all intent. In this case the task of intent handling would be to reach a state that is as close to the ideal state as possible.

This discussion also indicates that intent fulfillment is related to the measured state of the system. The goal of intent handling is therefore to close the gap between the measured current state and the wanted ideal state.

A system with the capability to act proactively would be able to predict future states. This allows it to detect intent degradation ahead of time and avoid it by acting with preventive measures before a non-preferential situation is actually happening. In any case intent is the decisive factor for determining if a current or future state is preferential or considered degraded. However, when dealing with predicted future states there is always a considerable amount of uncertainty. Also, future intent changes might require re-evaluation of the situation and change of planned actions.

# 19. Appendix A: Understanding RDF/RDFS/OWL modeling and reading TURTLE notation

## 19.1. The RDF modeling stack

W3C has specified a set of models targeting use case in knowledge engineering and rooted in symbolic AI methods and enabling inference through logical reasoning:

- RDF: Resource Description Framework [rdf], [rdfprim], [rdfsem]
- RDFS: Resource Description Framework Schema [rdfs]
- OWL: Web Ontology Language [owl], [owl2doc]

The models can be expressed with multiple notation formats. W3C describes the following:

- TURTLE: Terse EDF Triple Language [turtle]
- RDF/XML: XML notation for RDF [rdfxml]
- JSON-LD: JavaScript Object Notation for Linked Data [jsonld], [rdfjson]

The IG1253 family of documents uses TURTLE notation for modeling examples, because it provides the most intuitive and readable syntax.

All related specifications by W3C can be found at https://www.w3.org/TR/?tag=data.

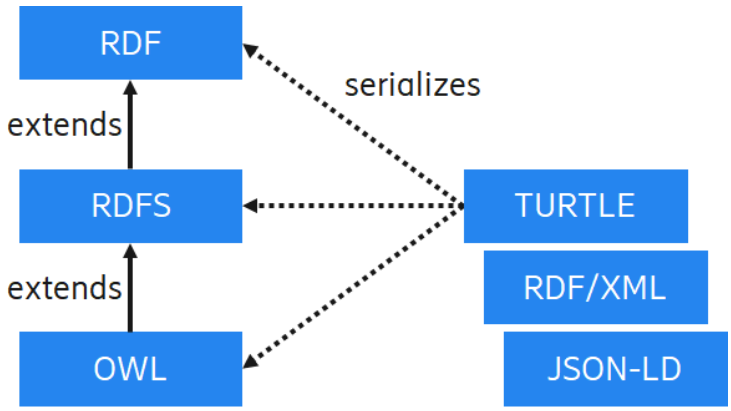


**Figure 19-1: Overview of RDF modeling stack and notation formats**

## 19.2. Triples as basic building blocks

In RDF models everything is expressed as triple statements in the form of
SUBJECT PREDICATE OBJECT

An RDF model consists therefore of a set of triple statements.

Predicates are often also referred to as properties. This means that the object is considered to be a property of the subject.

Figure 19-2 shows the equivalent graph representation of the triple statement.

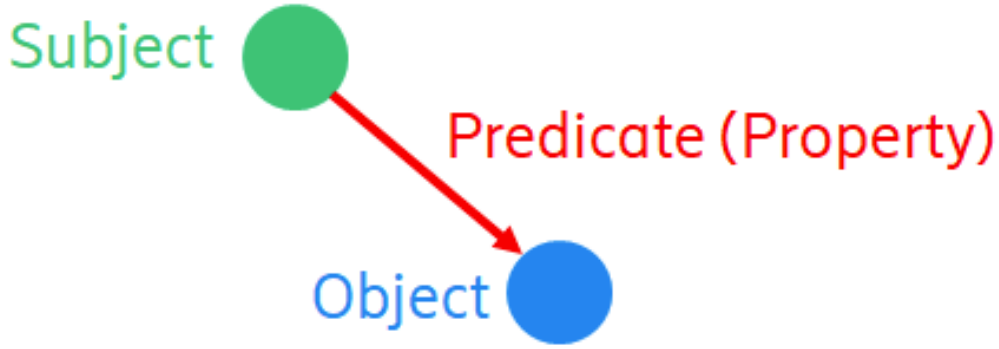


**Figure 19-2: Graph representation of triples as knowledge graph**

Subjects and objects are nodes in the graph and predicates establish a link between them. Different predicates have different meanings and semantics associated with them. Using a particular predicate within a triple statement establishes the knowledge that the subject and object have the respective relationship. Any number of predicates can be defined and used for expressing relationships.

Please note that the subject of one statement can be the object of another one. This way entire knowledge graphs are built with triple statements as basic building blocks. Different predicates are used to establish knowledge about different types of relationship between the nodes in the graphs. Figure 19-3 shows an example graph with multiple predicates and further nodes. Every predicate arrow would correspond to one triple statement in the textual notation of the model. The arrow represents the predicate pointing from the subject to the object of the triple statement.

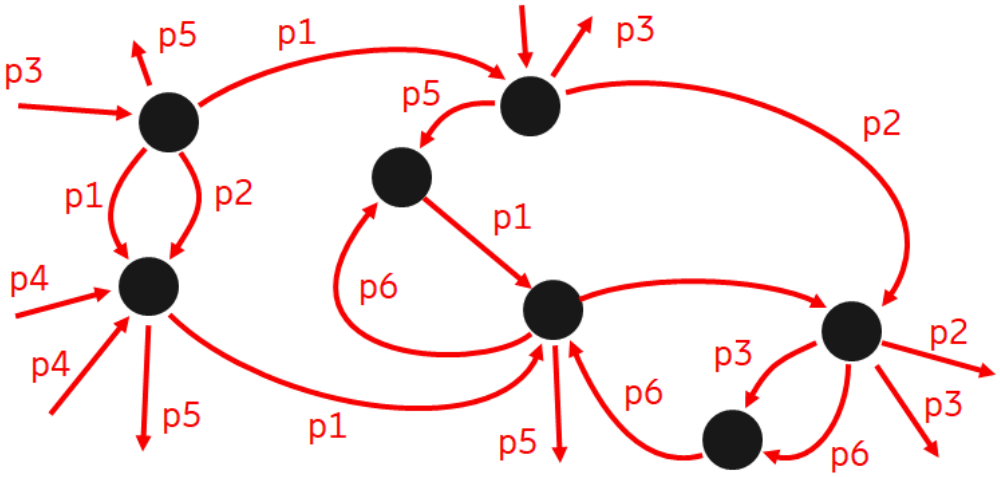


**Figure 19-3: Knowledge graphs using multiple predicates**

## 19.3. Referencing by IRI/URI

Everything in an RDF based model is referenced by an IRI or URI.

**URI: Uniform Resource Identifier [uri]:**
URIs are a sequence of characters that identify a physical or logical resource. It can be used to identify anything, such as real-world objects, people, places, concepts, information resources, …
They are standardized by IETF in RFC3986.

**IRI: Internationalized Resource Identifier [iri]:**
Defined by IETF in RFC3987 and greatly expanding the set of characters permitted in URI.

Subjects, predicates and objects are referred to by an IRI/URI. This means a subject-predicate-object statement in RDF takes the following form:
`<IRI of subject> <IRI of predicate> <IRI of object>`

For example:

```
http://example.com/IntentIndividuals/intent00001   http://www.w3.org/1999
/02/22-rdf-syntax-
ns#type  https://www.tmforum.org/2020/07/IntentCommonModel/Intent .
```

This example expresses that "intent00001" in the example.com domain is of type intent. The "type" property used is the type according to the RDF standard and "Intent" is a class defined in the TM Forum intent common model. This statement expresses that "intent00001" is an individual of the class "Intent". Or in other words, "intent00001" is an intent.

This chapter uses green, red, and blue colors in examples to clarify what the subject predicates and objects are within the statements. The color coding is not part of RDF standard and has no meaning for the models.

When multiple statements are given there is no meaning associated with or implied by their order.

Through rigorous use of IRI/URI, models based on RDF/RDFS/OWL have globally unique references for everything they model. This is the foundation for combining and federating models.

## 19.4. TURTLE makes RDF models readable

TURTLE is a notation format for RDF based models. It has a textual representation that provides a syntax that is more intuitively readable by human users.

The bare RDF statement using full URI are hard to read due to the use of IRI/URI. Much of an IRI/URI string is however repeated many times within a model and typically only the final part of the string is in focus of modeling. TURTLE allows to define a prefix representing a part of a IRI/URI string.

The following example uses turtle notation for the example statement given in the previous chapter:

```
@prefix    : http://example.com/IntentIndividuals/ .
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns# .
@prefix icm: https://www.tmforum.org/2020/07/IntentCommonModel/ .

:intent00001 rdf:type icm:Intent  .
```

Here three prefixes are defined in the TURTLE document to be used within all statements. By using the prefixes in the example statement about the type of the intent individual it became much more readable. Prefixes can also be understood as distinguishing namespaces used in the model.

Statements in TURTLE syntax are terminated by a full stop "." character. It is naming convention and common practice to start the names of classes with an upper case letter and the names of predicates.properties with a lower case letter.

TURTLE has a reserved keyword "a". It represents a shortcut for the often needed "rdf:type". The following statements are therefore equivalent:

```
ex:intent00001 a icm:Intent .
ex:intent00001 rdf:type icm:Intent .
```

It also allows to use the keywords "true" and "false" instead of the boolean individuals "xsd:true" or "xsd:false" from XML Schema.

## 19.5. Nature of objects

Everything that can be referenced by an IRI/URI can be an object in RDF statements. This can be objects in the same knowledge base, but due to the global referencing it can be any object globally. For example, the intent residing within the knowledge base of the intent manager can reference a service within a catalog or a resource individual within an inventory as long as a URI can be created that addresses these objects in the respective systems. In many cases URI can be created and used even if the targeted system is not using RDF based information models.

Objects can also be literals, such as strings, integers or boolean values. For example:

```
ex:intent00001 rdfs:comment "example intent used as example" .
ex:SliceLatency ex:latestMeasurement 10 .
```

XML Schema datatypes are typically used to explicitly state the type of literals:

```
ex:SliceLatency ex:unit "Second^^xsd":string .
```

Labelled blank nodes (b-node) are expressed by "_:" followed by its label:

```
ex:expectation01 icm:target _:x .
_:x rdf:type ex:Service .
```

In this example a blank node with label "x" is used. Labeled blank nodes can be subjects as well as objects in statements.

Blank nodes can be used as variables or placeholders, for example, if a concrete individual of the meant object is not known.

Next to labeled blank nodes there are also unlabeled blank nodes represented by square brackets "[ ]".

The following statements create the knowledge graph in Figure 19.4:

```
ex:expectation01 a icm:DeliveryExpectation .
ex:intent00001 icm:hasExpectation ex:expectation01 .
```

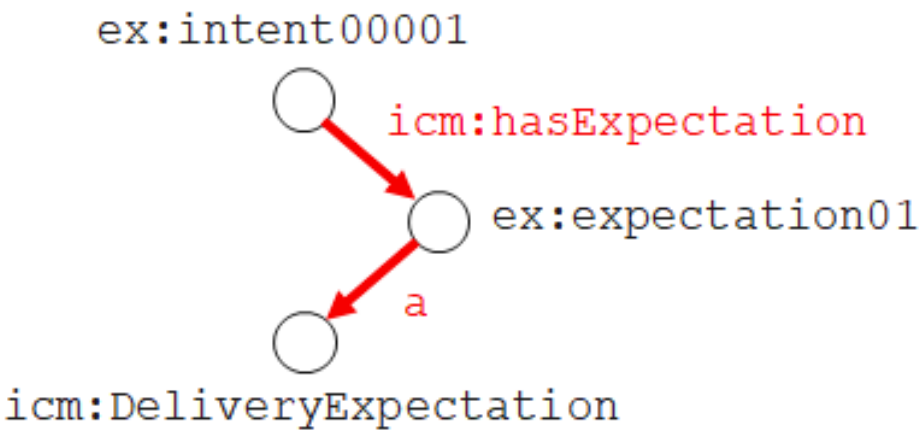


**Figure 19-4. Examples graph, where all nodes are labeled by IRI/URI.**

We can model a similar graph using unlabeled blank nodes and nested statements:

```
ex:intent00001 icm:hasExpectation [ a icm:DeliveryExpectation ] .
```

The graph in Figure B5 is equivalent to the one in Figure B4 with the only difference that the mode formerly referenced by ex:expectation01 has now no IRI/URI. It is however still an instance of class icm:DeliveryExpectation.

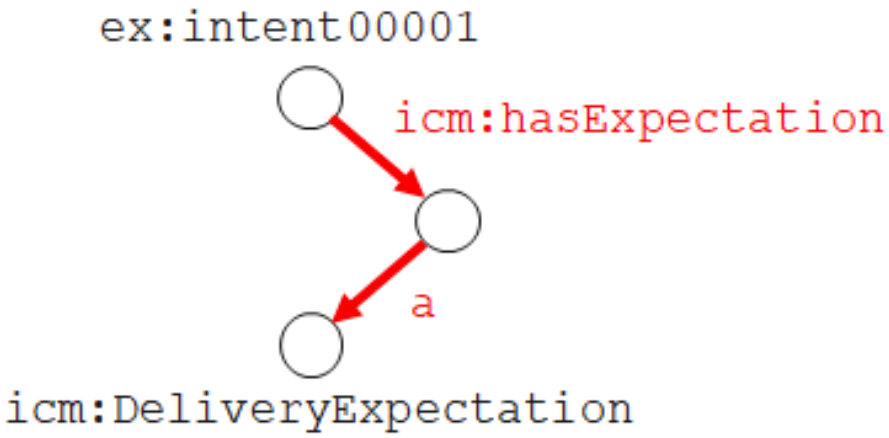


**Figure 19-5. Example graph with blank node**

The unlabeled node is represented by the square brackets "[ ]". There are still two distinct statements in the textual representation. The blank node "[ ]" is the object of the first statement with "icm:hasExpectation" as predicate. This same blank node is used as implied subject of all statements given between "[" and "]". In this example the statement "a icm:DeliveryExpectation" has the blank node as subject. It is expressing that the blank node is something of class delivery expectation.

The following example shows some more expressiveness involving unlabeled blank nodes:

```
[] foaf:knows [ foaf:name "Bob" ] .
```

This statement expresses "Someone knows someone else, who has the name Bob". Here the first blank nodes used as subject has no properties, and we therefore do not know more about it. The second blank nodes has the property foaf:name with the value "Bob". So we know, whatever or whoever this blank node represents, it has the name "Bob". This example uses the friend of a friend (foaf) ontology.

## 19.6. Predicate lists

Often multiple predicates reference the same subject:

```
ex:intent00001 a icm:Intent .
ex:intent00001 rdfs:comment “example intent” .
ex:intent00001 ex:somePredicate ex:SomeObject .
```

In TURTLE the same can be expressed by a series of predicates and objects, separated by a semicolon ";", following a subject. This means the subject only needs to be stated once. The semicolon symbol is used to repeat the subject subsequent triple statements:

```
ex:intent00001 a icm:Intent ;
               rdfs:comment "example intent" ;
               ex:somePredicate ex:SomeObject .
```

## 19.7. Object Lists

Often multiple triple statements have the same subjects and predicates with different objects:

```
ex:intent1 icm:hasExpectation ex:E01 .
ex:intent1 icm:hasExpectation ex:E02 .
ex:intent1 icm:hasExpectation ex:E03 .
```

In TURTLE notation the same can be expressed by a series of objects separated by comma "," and following a predicate. This means that the subject and predicate are repeated for each object.

```
ex:intent1 icm:hasExpectation ex:E01, ex:E02, ex:E03 .
```

## 19.8. Domain and Range in model definitions

When defining models, this involves a description of its vocabulary. Usually it defines classes, individuals, and properties/predicates. Properties are described with their domain and range:

- Domain specifies what can be used as subject in triple statement with this property
- Range specifies what can be used as objects in triple statements with this property

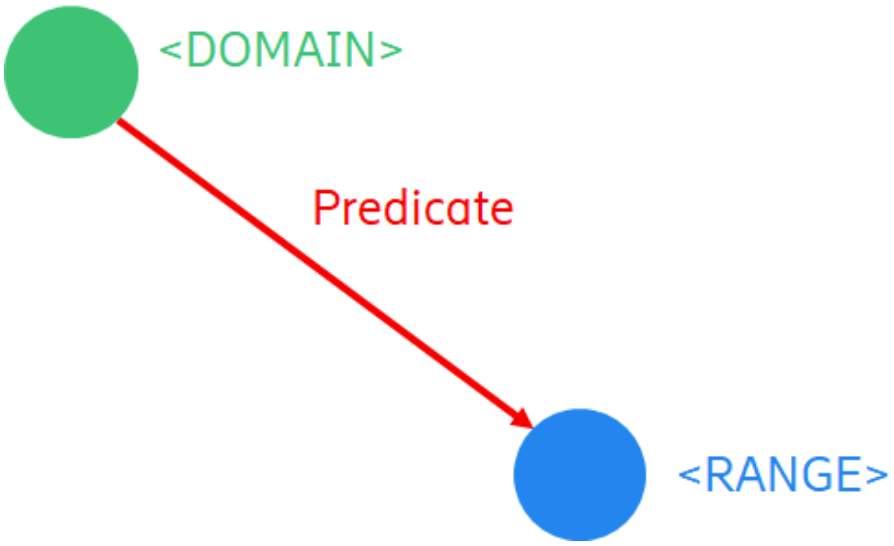


Figure 19.6: Illustration of domain and range of a predicate

For example, the property "icm:hasExpectation" is introduced in the intent common model in IG1253A. It has the domain of "icm:Intent" and therefore can be used as property of subjects of class icm:Intent. It also is defined with the range "icm:Expectation". This means in triples, this property has objects that are of class icm:Expectation. This means the property icm:hasExpectation can be used to link expectation objects to intents.

Blank nodes are used as subjects or objects in triple statements, and they contain statements with properties about them. It is possible to state explicitly the class of this blank node, for example:

```
ex:intent1 icm:hasExpectation [ a icm:Expectation
;                                 icm:target _:service ] .
```

In the following statement the class of the blank node is not explicitly given:

```
ex:intent1 icm:hasExpectation [ icm:target _:service ] .
```

However, the "icm:hasExpectation" property has a range definition of objects of class "icm:Expectation". When using the model it can therefore be inferred from the range of the property, that the blank node is of class "icm:Expectation".

There are however also Subclasses of the class icm:Expectations, for example icm:DeliveryExpectation. If the blank node is more specifically one of the Subclasses, this cannot be inferred implicitly, because there are multiple options and it needs to be stated explicitly:

```
ex:intent1 icm:hasExpectation [ a icm:DeliveryExpectation
;                                 icm:target _:service ] .
```

Please note that this does not violate the range specification because it covers all sub-classes as well.

# 20.Appendix B: Terminology

**Domain Information model**

Data used within an application is typically created by management function, stored in databases communicated over interfaces. Information models are used to describe the individual data elements and the structures to organize and categorize them. These information models are specific to an application domain in the same way as the data they model. In the expression of intent a typical task would be to define target values for domain specific data elements.

**Domain-specific model**

Intent-based operation is a generic paradigm of operating based on declarative knowledge about requirements rather than imperative invocation of operational processes. Intent is in this respect the carrier of knowledge of these requirements. The expression of intent is based on vocabulary and its semantics defined by models and standardized by TM Forum and other SDOs. The expressiveness needed is in many cases dependent on the application domain and system level where the intent is used and for which it needs to express requirements. The vocabulary and semantics needed to gain sufficient expressiveness for requirements within  a domain would be defined in domain-specific models. They are typically based on a domain independent model and provide extension and specialization to the generic concepts. The domain independent model for intent expression is referred to as intent common model and any model that builds on the intent common model by defining expanded vocabulary and semantics is referred to as intent extension model. Intent extension models are there for typically domain or use case specific.

**Intent Interface**

Intent and intent reports are carriers of information that would be communicated between two instances of intent management functions. The intent interface is the interface between two instances of intent management functions. They exchange information in the form of intent and intent reports over this interface. Over this interface all communication concerning intent and between intent management function is done. They manage the life cycle of individual intent objects, report on success and state, coordinate operational priorities, assess feasibility and negotiate what detailed requirement an intent can contain to successfully handle it and reach a compliant system state. We model this interface with the introduction of an intent handling management service. An intent management function in the role of intent handler would be the producer of the intent handling management service. Another intent management function in the role of intent owner would be the consumer of it.

**Intent Object**

An intent object is an individual intent. It can come in various forms and expressions. It can for example be a text document that uses a notation format such as TURTLE, XML or JSON-LD to encode all information of the intent. Such a document would also be used to communicate intent objects over an interface. Within a knowledge base an intent object is a node in the knowledge graph. It is an individual of class icm:intent according to the intent common model defined in IG1253 A. The detailed information and requirement specifications an intent typically consists of are contained in the properties associated with this individual node. The textual and graph representations of intent objects are equivalent and can be completely converted into each other.

# 21. Appendix B: Mapping the TM Forum Intent Model to other SDO Models

Intent is a specification of a list of expectations with each expectation containing the requirements, goals, and constraints to be achieved. Within an expectation, target instance and all the properties of this targeted instance can be defined. SDOs should keep in line with the intent common model firstly, then extent their domain intent model, and define their extension models.

## 21.1. 3GPP

The 3GPP [28.312] defines the structure of an intent as a list of expectations with each expectation containing objects and targets. The object is defined to contain 3 attributes - the *objectInstance*, *objectType* and the *objectContexts*. The *expectationContexts* describes the list of context(s) which represents the constraints and conditions that should apply for a specific expectation.

The attributes of 3GPP Intent Expectation can be mapped to the TM Forum model. The following table illustrates the mapping between TM Forum Intent Expectation and 3GPP Intent Expectation.

Table 3. The mapping between TM Forum Intent Expectation and 3GPP Intent Expectation

| **TM Forum Intent Expectation (TR290 v2.0.0 )** | **3GPP Intent Expectation ([28.312])** |
|---|---|
| **Attribute** | **Class Property** |
| icm:target | expectationObject.ObjectInstance |
| icm:propertyParams | expectationTargets |
| | expectationContexts |
| icm:deliveryParams | expectationObject.objectType |
| | expectationObject.ObjectContexts |

# 22.Appendix C: Abbreviations and acronyms

3GPP 3rd Generation Partnership Project
ANF Autonomous Networks Framework
BSS Business Support System
DN Distinguished Name
DSL Domain Specific Language
IETF Internet Engineering Task Force
IoT Internet of Things
IRI Internationalized Resource Identifier
ISO International Organization for Standardization
ITU International Telecommunication Union
JSON JavaScript Object Notation
JSON-LD JavaScript Object Notation for Linked Data
KPI Key Performance Indicator
LDAP Lightweight Directory Access Protocol
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 Ain't Markup Language

# 23.Appendix D: References

[28.312] 3GPP TS 28.312: "Management and orchestration; Intent driven management services for mobile networks, Release 17, 2022-5

[28.812] 3GPP TR 28.812: "Study on scenarios for intent-driven management services for mobile network, V17.1.0, 2020-12

[32.158] 3GPP TS 32.158, V16.04, 2021-09, Design Rules for Representational State Transfer (REST), https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3396

[32.300] 3GPP TS 32.300 version 5.0.2 Release 5, Name convention for Managed Objects, https://www.etsi.org/deliver/etsi_ts/132300_132399/132300/05.00.02_60/ts_132300v050002p.pdf

[dbpedia] DBpedia, Global and Unified Access to Knowledge Graphs, https://www.dbpedia.org/

[dc] Dublin Core Metadata Initiative Specifications, https://www.dublincore.org/specifications/dublin-core/

[dcat] Data Catalog Vocabulary DCAT, Version 3, W3C Working Draft, 04 May 2021, https://www.w3.org/TR/2021/WD-vocab-dcat-3-20210504/

[foaf] FOAF, Friend of a friend ontology, Namespace Document, 14 January 2014, http://xmlns.com/foaf/spec/#term_LabelProperty

[ibn] A. Clemm, L. Ciavaglia, L. Z. Granville and J. Tantsura: "Intent-driven Networking - Concepts and Definitions"

[ig1218] Autonomous Networks Business Requirements and Framework v2.1.0

[ig1230] TM Forum IG1230: "Autonomous Networks Technical Architecture v1.1.0"

[ig1252] TM Forum IG1252, Autonomous Networks Levels Evaluation Methodology v1.2.0

[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

[iri] Internationalized Resource Identifiers (IRIs). M. Duerst; M. Suignard. IETF. January 2005. Proposed Standard, RFC3987. URL: https://tools.ietf.org/html/rfc3987

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

[ocl] Object Management Group, "OCL 2.0 Specification" OMG Specification, June 2005

[owl] OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation, 11 December 2012, https://www.w3.org/TR/owl2-primer/

[owl2doc] OWL 2 Web Ontology Language Document Overview (Second Edition), W3C OWL Working Group, 11 December 2012, https://www.w3.org/TR/owl2-overview/

[owltime] Time Ontology in OWL, W3C Candidate Recommendation. 26 March 2020, https://www.w3.org/TR/2020/CR-owl-time-20200326/

[prof] The Profiles Vocabulary, W3C Working Group Note, 18 December 2019, https://www.w3.org/TR/2019/NOTE-dx-prof-20191218/

[raci] M. Smith, J. Erwin: "Role & Responsibility Charting (RACI)", Project Management Forum, 2005

[rdb2rdf] A Direct Mapping of Relational Data to RDF, W3C Proposed Recommendation 14 August 2012, https://www.w3.org/TR/2012/PR-rdb-direct-mapping-20120814/

[rdb2rdf_uc] Use Cases and Requirements for Mapping Relational Databases to RDF, W3C Working Draft 8 June 2010, https://www.w3.org/TR/2010/WD-rdb2rdf-ucr-20100608/

[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/

[rdfsem] W2C, RDF Semantics, W3C Recommendation 25 February 2014, https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/

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

[rfc7575] M. H. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. E. Carpenter, S. Jiang and L. Ciavaglia: "Autonomic Networking: Definitions and Design Goals (RFC7575)"

[select] Selectors and States, W3C Reference Note: https://www.w3.org/TR/2017/NOTE-selectors-states-20170223/

[shacl] Shapes Constraint Language (SHACL). Holger Knublauch; Dimitris Kontokostas. W3C. 20 July 2017. W3C Recommendation. URL: https://www.w3.org/TR/shacl/

[shex] Shape Expression Language 2.next (SHEX), 2019-08-31. W3C Community Group Draft Report, https://shexspec.github.io/spec/

[skos] SKOS Simple Knowledge Organization System Reference. Alistair Miles; Sean Bechhofer. W3C. 18 August 2009. W3C Recommendation. URL: https://www.w3.org/TR/skos-reference/

[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/

[uml] Unified Modeling Language Specification, Version 2.0, July 2005, Object Management Group, https://www.omg.org/spec/UML/2.0/

[umlowl] A detailed Comparison of UML and OWL, Kilian Kiko and Colin Atkinson, University of Mannheim, 2008,

[uri] Uniform Resource Identifier (URI): Generic Syntax, T. Berners-Lee; R. Fielding; L. Masinter. IETF. January 2005. Internet Standard RFC3986, https://datatracker.ietf.org/doc/html/rfc3986

[w3cdocs] W3C, All Standards and Drafts, https://www.w3.org/TR/?tag=data

[x500] ITU-T Recommendation X.500 (1993): "Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services". https://www.itu.int/itu-t/recommendations/rec.aspx?rec=2996&lang=en

# 24. Appendix E: Future work

The following topics need some further documentation and examples and will be addressed in future phases of the Autonomous Networks project.

| Topic | Description |
|---|---|
| End-to-end use cases | The document IG1253E is planned to be added in future releases. This guide will use the techniques and concepts from IG1253 and its sub-documents to show end-to-end autonomous operation. |

# 25.Administrative Appendix

## 25.1. Document History

### 25.1.1. Version History

| Version Number | Date Modified | Modified by: | Description of changes |
|---|---|---|---|
| 1.0.0 | 28-May-2021 | Alan Pope | Initial Release |
| 1.1.0 | 26-Nov-2021 | Alan Pope | Minor updates |
| 1.2.0 | 31-Mar-2022 | Alan Pope | Updated to align with change from IG to TR documents |
| 1.3.0 | 23-Jun-2022 | Xie Yuan | Added Appendix B: Mapping the TM Forum Intent Model to other SDO Models |
| 1.3.0 | 01-Aug-2022 | Kevin McDonnell | Changes to Executive Summary section. Proofing. |

### 25.1.2. Release History

| Release Status | Date Modified | Modified by: | Description of changes |
|---|---|---|---|
| Pre-production | 28-May-2021 | Alan Pope | Final edits prior to publication |
| Production | 26-Jul-2021 | Adrienne Walcott | Updated to reflect TM Forum status |
| Pre-production | 26-Nov-2021 | Alan Pope | Updated to v1.1.0 |
| Production | 21-Jan-2022 | Adrienne Walcott | Updated to reflect TM Forum Approved status. |
| Pre-production | 31-Mar-2022 | Alan Pope | Updated to v1.2.0 |
| Production | 20-May-2022 | Adrienne Walcott | Updated to reflect TM Forum Approved status. |
| Pre-production | 05-Aug-2022 | Alan Pope | Updated to v1.3.0 |
| Production | 23-Sep-2022 | Adrienne Walcott | Updated to reflect TM Forum status |

## 25.2. Acknowledgments

### 25.2.1. Guide Lead & Author

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

### 25.2.2. 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 |
| Xie Yuan | Systems Expert | Huawei |

### 25.2.3. Additional Inputs

| Member | Title | Company |
|---|---|---|
| Lester Thomas | Chief IT Systems Architect | Vodafone Group |
| Ankur Goyal | Lead Consultant | Infosys |
| Emmanuel A. Otchere | Chief Technical Expert, VP Standards & Industry Development | Huawei |
| Min He | Chief Architect | Futurewei |