TM Forum Introductory Guide

Intent Based Operation User Guide

IG1358

Maturity Level: **Alpha**	Team Approved Date: 03-Sep-2024
Release Status: Pre-production	Approval Status: Member Evaluated
Version 1.0.0	IPR Mode: RAND

Notice

Direct inquiries to the TM Forum office:

181 New Road, Suite 304
Parsippany, NJ 07054, USA
Tel No. +1 862 227 1648
TM Forum Web Page: www.tmforum.org

Table of Contents

List of Figures

Executive Summary

This user guide explains the concepts and modelling of intent based operation as specified by TM Forum. The content of the user guide is available as a downloadable document as well as presented on TM Forum's website dedicated to the TM Forum Intent Ontology (TIO).

Intent based operation is a key component of autonomous networks. The TM Forum technical architecture includes intent management within autonomous domains. Open APIs were updated to allow communicating intent and intent reporting and manage the content of intent. This includes intent being used as part of service management. This is based on the TM Forum intent ontology providing the vocabulary for expressing intent and intent reports. This user guide explains these concepts and how to apply the proposals in practical use cases.

1. Introduction

Autonomy in networks means that business, service, and resource operations are able to adapt decisions and actions dynamically to changing goals and requirements and cover a broad range of situations without human interventions. In this new environment, the business objectives of the Communication Service Provider (CSP aka Network Operator) as well as expectations of customers and users need to be communicated to the software that constitutes an autonomous network. This is the role of intent. Intent describes the business objective of the CSP as well as the expectations of customers and users. In this respect, intent establishes machine-readable knowledge about goals, targets, requirements, and constraints.

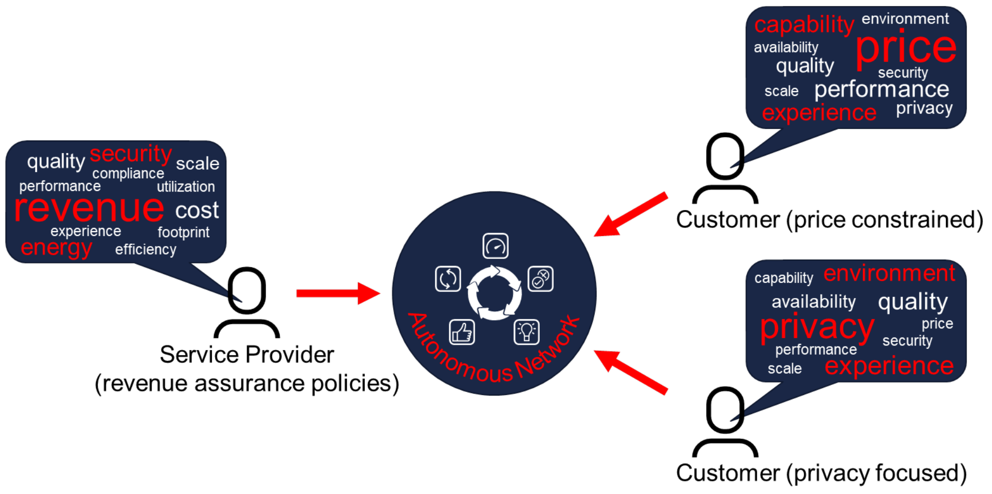

quality security scale compliance performance utilization revenue cost experience footprint energy efficiency
Service Provider (revenue assurance policies)
Autonomous Network
capability environment availability price quality security scale performance experience privacy
Customer (price constrained)
capability environment availability quality privacy price security performance scale experience
Customer (privacy focused)

Figure 1-1. Intent of CSPs and their customers

Autonomous networks serve multiple parties with different interests and preferences. This is illustrated in Figure 1.1. The CSP's main objective is to make a revenue from offering network based services. It requires the network operation to be cost-effective and secure. Some of its customers are very price sensitive. They want reasonable service experience and service availability for a competitive price. Other customers value privacy. A major factor in their experience would therefore be confidential use of information and features such as encryption. Customers with use cases, such as gaming and remote driving require consistent low latency, while customers with video applications also need reliable high throughput. Intent is primarily the information object used by all these parties for expressing their needs and requirements towards the autonomous network. Furthermore, intent is also the means by which subsystems within the autonomous network communicate requirements to each other.

In this respect, intent defines what an autonomous network is expected to achieve, but it leaves the details of how a network is designed and operated to the internal operations of the network platform. This means that the smart software on the platform can constantly optimize how the service is delivered, and we can incrementally add new technologies like Analytics and Machine Learning to constantly improve the implementation. For autonomous networks, the Intent-interaction model is a must, not optional. The role of Intent is beyond relieving the burden of the user knowing implementation details. More importantly, it sets the autonomous system's internal goal. And the system then takes proactive actions to achieve the stated goal based on its observation of the environment. So Intent is the fundamental mechanism to utilize the service of autonomous networks.

1.1. Scope of the intent user guide

Intent based operation is a key component of autonomous networks. The TM Forum technical architecture includes intent management within autonomous domains. Open APIs were updated to allow communicating intent and intent reporting and manage the content of intent. This includes intent being used as part of service management. This is based on the TM Forum intent ontology providing the vocabulary for expressing intent and intent reports. This user guide explains these concepts and how to apply the proposals in practical use cases.

This user guides explains the concepts of intent-based operation as integral part of autonomous networks. It demonstrates how the TIO can be used to express intent for a variety of use cases and how the APIs shall be used. The user guide will expand over time covering advanced features such as intent negotiation, security, conflict resolution and the interaction with other SDOs.

The content presented in this user guide explains the recommendations regarding intent-based autonomous networks and the TM Forum intent Ontology that was introduced by various introductory guides, technical reports and APIs. Notable contributions and recommended further references are:

- Autonomous Networks Technical Architecture [ig1230-ta]
- Autonomous Networks - Reference Architecture [ig1251-ra]
- Intent in Autonomous Networks [ig1253-intent]
- The technical reports specifying the models for intent expression in the TR290, TR291, TR292, TR293, TR294 and TR299 series of documents.
- The open APIs with the intent API TMF291 and the version 5 updated APIs for product and service management with intent management capabilities TMF641, TMF645, TMF622, TMF679. [oda-openapis]

The intent user guide is based on these contributions and will explain their concepts and use for practical use cases. Starting with this initial version of the user guide, the content and covered use cases will be expanded over the following release cycles. The user guide will be published in pre-production state until the major features of the TM Forum intent ontology and the intent interface are sufficiently covered, and model artifacts are available.

2. Motivation for intent driven autonomous networks

There are typically two driving forces that fuel the evolution of networks: Improvement of customer experience and cost-effectiveness. In this respect the term autonomous networks corresponds to technologies used in network operation that addresses both.

Customer experience can be determined by multiple factors. A key component is the experience in usage of network-based services. Does the service provide sufficient performance for what the users want to do? Are all needed services always available? This means that customers are not limited in what they need to do and can rely on the services. This also indicates that customized services and experience that specifically address individual needs can be a distinguishing factor for the service offers of communication service providers. This concept also extends beyond the direct experience of human end users. In many cases the consumer of network services today is an automated process or device. Typical use cases range from low power IoT devices with low volume and occasional data transmission needs to connected cars and autonomous drones needing reliable high bandwidth connectivity also meeting real-time requirements. Another example of demanding high volume scenarios are video broadcasts and support of events with many users in a single location for a limited time. These are only a few examples of use cases. An autonomous network is supposed to deliver the right service experience in all these scenarios although the demands and challenges to address may be very different. An autonomous network is expected to improve the customer experience by introducing automated discovery of current and future issues in network and service operation followed by an automated taking of actions. This means an autonomous networks profits from 24/7 attention and reaction with extremely low latency compared to processes that involve human attention and actions. Therefore, an autonomous network will be able to minimize the time a Customer might be exposed to service and experience degradation, or even avoid degradation altogether through early identification of situations. An autonomous network will therefore have the capability of proactive healing and continuous optimization of customer experience.

Cost-effectiveness of network operation considers operational and capacity costs. A major factor in operational cost is the human workforce needed to monitor the operation at scale and execute operational tasks. As long as human personnel is directly involved in operational processes the workforce needs scales with the complexity of the network and the offered services. This means, networks with high complexity and diverse service offers would require a huge human workforce unless automation and the degree of autonomy is significantly increased. This is a key target of autonomous networks: making the human workforce more efficient by increasing the ratio of operational tasks per workforce size. This does not necessarily mean a reduction of human workforce, but a more efficient use of their competence. It will shift from direct operational tasks towards oversight and design of more efficient automated and potentially AI driven processes. AI-Ops and ML-Ops processes support the data scientist in generating efficient AI based processes that allow adaptive autonomous operation. The use of intent shifts the dialog with the autonomous network from tasks a human requires to be executed to results to be achieved. It therefore transitions from humans needing to decide what is done and only utilizing automated execution of their decisions, towards communication of expected outcomes leaving the decision how to achieve this outcome and what actions to execute to the autonomous systems for service and network operation. This offloads a huge amount of work from the human workforce to the systems that operate the network. The use of large language models on the human machine interface further improves efficiency through natural language based and therefore intuitive interaction with the autonomous functions within network operation.

Capacity costs are mainly driven by the amount of equipment and resources needed to deliver services to customers. It naturally scales with the size of the network in terms of geographical coverage, number of users and number and complexity of services. Here efficiency of resource usage means to deliver more services and serve more users with as small amount of resources as possible. Autonomous networks will be able to continuously monitor resource usage versus user needs and optimize at a scale and frequency that would not be possible without autonomous operation. It will also help to identify critical bottlenecks where capacity investments are most effective.

These use cases illustrate how autonomous networks can be a key differentiator for CSPs. Intent is a central concept in the realization of the mentioned capabilities. TM Forum has published a huge body of work regarding intent. There are intent aware open APIs, such as the dedicated intent API TMF921 as well as many of the product and service management APIs, such as and product and service order APIs TMF622 and TMF641 or product and service qualification APIs TMF679 and TMF645. Intent specification is introduced in product and service catalogs. Furthermore, TM Forum has published the TM Forum Intent Ontology (TIO). It is a system of information models to be used for expressing intent and intent reports in the communication of intent through the intent aware APIs.

2.1. The zero-touch paradigm and role of intent

An autonomous network is a zero-touch network. This means, that ultimately there are no human actions needed any more to operate it. Or in other words, nobody has to touch it. This is the ultimate goal and till constitute level 5 autonomy with full and self adaptive automation.

The role of the human workforce shifts to oversight. Humans define business strategies and then describe to the network what they expect it to achieve in order to support the strategy. This indicates already what the role of intent and intent based interfaces are. They provide the means by which the requirements the network is expected to meet are expressed and communicated.

To better understand the conceptual changes of autonomous and intent-driven network operation it is helpful to compare several evolution stages starting with human operated networks and ending at future fully autonomous networks. Autonomous operation will be the result of a transition journey from manually operated networks through automation towards self-adaptive autonomous networks. In this journey the level of autonomy is continuously increased.

2.1.1. Tasks in network operation

There are two distinct tasks in network operation regardless if it is manually operated by humans or autonomously operated by automated systems: A decision needs to be made, which determines what to do, and then this decision needs to be implemented through execution of actions. This is also valid for any layer of operation regardless if we consider business, service, or resource operation layer.

In network operation a situation would be identified and classified as problematic or not based on observation, knowledge, and experience. Then possible solutions are created and evaluated with respect to their expected outcome. This involves considering the solution's direct impact on the network performance, service delivery and user experience. For example, changes in service performance and availability are direct consequences of an operation action. An action that solves a user experience issue with improvements of service performance therefore appears preferential. However, a good decision also considers collateral effects. An action that solves a problem, but creates another even bigger one would not be suitable. Also indirect effects that impact the business outcome and long term strategy need to be considered. For example, the cost of a new network configuration that improves user experience need to be evaluated with respect to its potential increased resource usage or degradation of SLA resulting in penalties.

Further considerations are robustness and risk. An operational action might have a likely preferential outcome, but if it also has a potentially small but not negligible chance to fail with catastrophic consequences, the risk of taking this action outweighs the potential gain. Other factors that might play a role are the available budget or the general strategy of the SCP. For example, does the CSP put increased focus on highly reliable customer experience or delivery of budget friendly standard services. Is there a need to improve carbon footprint or the profile of ethical organization. These examples show that solutions can lead to conflict that needs to be resolved by finding a balanced solution strategy. This means that decisions to do the right operational actions are complex. They potentially involve very diverse and not only technical considerations.

Once a solution strategy is found and decided, it needs to be implemented in the network. This typically involves issuing commands into a management system to change configurations or modify services.

2.1.2. Manually operated networks

In a manually operated network humans are involved in decision-making as well as task execution. Humans are therefore directly involved in all processes and control loops. Some processes might be semi-automated with systems that automatically deliver information to the human decision maker. There are, for example, alarm management systems and a measurement and analytics infrastructure that supports the human network operation team. This helps to identify problems faster, and it provides comprehensive situational awareness allowing for making the right decisions. However, it is ultimately the human who finds solutions and decides what needs to be done. In manually operated networks humans are also the main actor involved in the implementation of decided solutions and execution of respective actions. Also, here there might be automated or semi-automated tools that automatically execute a sequence of actions and report on the success, but they mainly provide convenience and efficiency in task execution as all distinct actions are still initiated by a human.

Humans are typically good at making balanced decisions. They foremost have diverse knowledge that exceeds knowing the state of the network. They know, for example, the budget available. Their management provides guidelines rooted in CSP strategy. So they know if cost or quality are a priority helping them to decide between alternative solutions. Humans also have gained experience on how a network would behave after they do an action. They can anticipate side effects and risk. Most importantly they can combine all this into a balanced decision. Replicating this level of intelligent decision-making is one of the key challenges of autonomous operation.

The main drawback of human operation is the volume of tasks a human workforce is able to handle. This does not scale up as a human workforce is a major cost factor as well. Its reduction and replacement has therefore a direct business motivation. Furthermore, humans are slow. The identification of problems, their analysis, finding of solutions and execution of actions takes time. This leaves the network and its users exposed to issues resulting in log overall network performance with respect to resource utilization and user experience. Therefore, any means to replace human driven operation processes with automated ones can have substantial effect on the business outcome for the CSP.

2.1.3. Process automation with policies and service abstraction

Some means of automation in network management and operation are already widely deployed. Two key concepts are combined: abstraction and policies.

The network infrastructure with its distinct resources is described in a multi-layer architecture. It abstracts the individual function and role of network resources and organizes them in layers and functional domains. The function and service a resource provides is treated separately from its implementation. Resources provide services through defined APIs and are organized through different layers of abstraction. This also means that replacing is transparent as long as the replacement provides the same services with the same capabilities. The network abstraction is done in multiple layers of network operation from network function and resource management, through end-to end service based operation to business operation layer that is concerned with products and service level agreements (SLA). This abstraction allows an CSP to market their network through defined products and services independent of the concrete implementation and composition of the network infrastructure.

Automation builds on these conceptual foundations. It is realized by automatically executed processes for concerns such as product ordering, service orchestration, service assurance and optimization among many others. The main technical means for achieving automated operation are policies. Policies constitute sets of rules. They are invoked either through explicit orders over an API or by pre-defined and actively monitored trigger criteria. This way, the automated system can detect and react to errors, anomalies or any situation that is observable from data. A policy is embedded in a framework and environment with automated measuring and analytics tools. The policy itself implements decision-making. Once active, a policy executes through a series of branching conditions until it arrives at a decision for actions to execute.

Although policies automate decisions, it is important to note that the intelligence behind the decision is still a human. The main task of a human policy developer is to predict situations the network operation might face and program the correct reaction into the policy. This involves a definition of how a situation can be detected from data to define trigger criteria. Furthermore, the policy developer determines what data needs to be consulted in the evaluation process for finding the right branch in the policy's decision tree. Executing these decisions will ultimately arrive at the planned correct system reaction, which then will be executed. In this respect the policy is mainly a recipe that represents human knowledge and intelligence ready to be automatically followed.

What we need to understand about policy driven automation is that the operations system can execute the human determined analysis and actions, but it usually does not have any means to understand why it is doing these actions. The system has no notion of why a certain system state made the policy developer prefer one action over another. It does not understand all requirements and knowledge available to the developers or their motivations to consider some actions to preferable and correct while others shall be disregarded. The system can only trust that the policy is correct and execute the actions it prescribes. This automation fails if it encounters new situations that were never considered by policy developers and therefore no respective rules were created to handle them. If this happens, the worst case scenario would be that the automated system wrongly applies one of its policies to a situation it was not designed for. The best scenario is that the system goes into a suboptimal, but safe state and informs the policy developers about the situation. A human can then assist and partially take over until eventually the policy gap is closed.

New situations like this are not always happening by surprise, if the CSP introduces new products or custom variations, this usually requires that operation policies need to be updated. As this is a human driven task it comes with a significant delay.

The use of machine learning for creating models that replace human developed policies does only marginally improve the automation capabilities. In typical machine learning based environments the intelligence lies in the creation of cost and reward functions designed by data scientists. They determine the wanted system behavior and expected results. The actual training process is a statistics based algorithm that finds model configurations based on precedence shown in historical data or simulation. Data scientists therefore take the same role as policy developers as the intelligent entity that determines the right decisions. Human designed and machine learned policies therefore differ in the way the policy and model is generated, but not in its nature as automatically executable but ultimately human determined solution.

In these automation environments there is often already a notion of variable goals and declarative requirements. For example service level objectives used within service management specify target values to be reached. This introduces degrees of freedom and variation to policies. It can activate certain branches the decision tree leading to adaptive behavior of the automated system. In machine learning based models further dimensions in the model's feature vector can be introduced. They make the model covering a greater variety of demands. However, this also increases model complexity resulting in computationally more demanding training. It also requires training data that consolidates and correlates more concerns and features. This complexity limits how much flexibility can be built into the automation processes based on these models.

Within the multi layered architecture, partial decisions are made on every layer. Policies throughout the system are modular components specialized on partial decisions. A correct end-to-end network behavior therefore requires that policies work well together across layers and domains. This is challenging to achieve consistently and reliably in today's complex networks that need to provide a great variety of different services. Typically, this is ensured through extensive testing. In policy logic is designed that uses knowledge about the policies in other layers and domains in their decision processes. This often creates extensive dependencies and tight coupling between layers.

This discussion leads to the following conclusion: While automated processes based on human designed or machine learned policies can achieve an impressive level of automation, the automated systems have limited capabilities to adapt. They need extensive human action and involvement humans to be kept up to date. The main reason for this is that policy decision are still human decisions requiring human knowledge and intelligence. Human involvement might be indirect, but it is still essential for ensuring correct operation. The analysis of system behavior and manual intervention to adapt the system's decision-making and action taking processes are still frequently needed. This binds a substantial and costly workforce.

2.1.4. The future of adaptive autonomous operation

The ultimate autonomous network would be told by a human what it shall achieve, and then the network chooses all needed actions fully autonomous. If it is consistently and reliably doing this correctly in all possible situations the network is fully autonomous at level 4 or 5. The analysis of policy-driven automation did show that fully automated decision-making requires a considerable evolution of networks operation concepts beyond policy driven processes. This also means that autonomy levels 4 or 5 are hard to achieve purely by scaling established tools and processes.

The main missing ingredient is a deeper understanding by the network operation system what it is expected to achieve and why this is needed. Why is some result good, and why is another result bad? What are the consequences of decisions and actions? This would allow the operating system to reflect about its current policies and find better ones. Such a system can adapt itself autonomously to changes in demand and expectation. The autonomous system's only interaction with the human workforce would be that humans keep the system up to date regarding requirements, preference and strategy. While the system is fully autonomously developing its own solutions and policies. This system has then become the intelligent entity that makes operational decisions.

Such a system is challenging to create even with latest cutting edge technologies available at the time of writing this analysis. There will be a gradual evolution of capabilities, which is reflected by an increase of autonomy level and reduction of direct and indirect human involvement in operational decisions. There are however a couple of ingredients that will facilitate this journey:

Intent provides knowledge to an autonomous system. It states the requirements expected to be met. Furthermore, it will be able to communicate preference and further relevant information about the needs and strategies of the CSP and their customers. An important aspect intent can help with is the knowledge about the effects of operational goals on business outcome. This allows us to make operational decisions based on maximizing the potential business outcome. These capabilities will make intent a central concept for achieving higher levels of autonomy. Intent can be used directly on the interface between the human and network, but also between autonomously operating subsystems allocated in autonomous domains. The work on the TM Forum Intent Ontology (TIO) and intent APIs has the goal to give intent information models these capabilities.

Intent also transitions the interaction between systems from imperative ordering of exposed processes and services to communicating declarative requirements. This property of intent and intent based interfaces marks an increase in abstraction. It particularly avoids that detailed knowledge about a systems action taking would be used across domains. It strengthens separation of concerns between autonomous subsystems. As such it is essential for keeping an autonomous network modular.

Policies will still be an important tool. On one hand a human designed or machine learned policy that correctly operates within its environment and scope is an asset and investment that should be used. The difference to previous generations of automation will be that a fully autonomous network has the capability to determine if the decisions a policy makes and the actions it takes are correct. If not the autonomous network can choose a policy that is better suited for the demands, notify the policy developer to make the needed adaptations, or it can itself define and create a better policy.

Machine learning based processes and models will be essential. They can determine the correct policies as long as sufficient and significant data is available. Here the autonomous network will be able to change the goals of the ML process and adapt the models. Key challenges are the broadness of domains with respect to features and concerns as well as the delay of repeated training. Intent expresses requirements and goals and as such it directly impacts cost and reward used in training.

Any other techniques aiming at analysis and decision-making has the potential to further improve autonomy. Examples are machine reasoning based on knowledge graphs for preserving and organizing domain knowledge. Probabilistic analysis can help to predict outcomes and assess risk. AI planning can identify the correct sequence of multiple actions for reaching the wanted outcome. Many more existing of future technologies will need to be researched and integrated into the solutions.

Another concern that can increase the operational efficiency of autonomous networks is the intuitive interaction with humans. The use of natural language removes barriers. This is a central use case for large language models in autonomous networks.

Improved data management is also essential. Every operating system is only able to control if it has the means to observe what it controls. Autonomous systems need to adapt, which also applies to their data management. It needs to deliver insights that match the expression of requirements.

3. Definition of Intent and its evolution

The term intent appeared in telecommunication and IT already in 2014 as “intent-driven networking” in the context of Software-defined Networks (SDN). Consequently, many early implementations of intent appeared in SDN controllers in 2015 and 2016. The early use of intent in SDN was very in a technical network context and was directly involved in selecting alternative policies. The intent was either initiating processes or it was used in branching within decision trees while provisioning and managing SDN assets. The definition of intent at the time reflected this use of intent. For example, in 2015 IETF states that intent is “An abstract, high-level policy used to operate the network.”

In the meantime, the understanding of intent as a concept and its role in automation has evolved from the policy and rule’s centric view towards being a universal entity for declaring and communicating goals, requirements, and constraints. The direct link to implementation assets such as policies and processes becomes less eminent with the tendency to be explicitly excluded. Furthermore, the setting of intent is often directly linked to human expectations and choices. In this respect, IETF has presented a new definition of intent in 2020, now intent is defined to be: “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.”

This example of a shift in definition can be observed throughout literature and standardization. The speed of this transition might vary, but a common direction is very evident.
In 2019 a technology assessment by Ericsson research did a broad overview of usages of intent throughout the industry and academic publications. It summarized the findings in a new definition: “[…] an intent is a declaration of business goals, operational goals or states that a system should meet, without specifying how to achieve them (i.e., not prescribing events, conditions or actions).”
The examples of newer definitions introduce the idea that intent is a declaration of goals independent of their implementation. Furthermore, it expands the scope of what intent describes and where it is used. Business and higher operational levels are in scope rather than just the network.

This newer definition of intent is also in line with the description of intent presented by ETSI ZSM:

> “Today intent-driven systems refer to intelligent software able to understand the user goals and translate automatically into a concrete prescription of service or network configuration.”
>
> “Intent is understood to be the information of knowledge objects describing these goals.”

From the perspective of the user, these goals express the expectation the user has directly with respect to the actions and behavior of the system. They might also formulate derived instrumental goals needed to fulfill human expectations.
The AN project definition follows the trend visible in the evolution of the intent concept and defines intent as follows:

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

This definition of intent focuses on the user perspective. The user has expectations, which are directly or indirectly expressed as goals, requirements, and constraints. This also means that the intent formalism must be capable of expressing all semantics involved when users express their expectations that are relevant for correct operation for an automated zero-touch domain.

These formally modeled and common semantics allow a technical system to understand what it is expected to do. It also implies that intent is encapsulated in self-contained objects, which can be exchanged over APIs to communicate the user expectations and operational expectations.

The definition also points to completeness. All expectations are formulated by intent. From the system perspective, this means that all it needs to operate and optimize against is the set of intent given to it and nothing else.

3.1. Other SDOs and how they define intent

Next to TM Forum many SDOs have started to work on intent. They have specific scopes of work and this might lead to differences in concepts and definitions. The different approaches were studied by TM Forum and are explained and summarized in [ig1259-approaches]. This chapter will be updated in future releases of the intent user guide to provide further and updated information regarding the work on intent in other SDOs and how it compares with and complements the proposals of TM Forum.

4. Categorization of Intent

Intent is a representation of requirements and intent based operation is therefore mainly concerned with life-cycle management of the requirements and finding solutions that would reach compliance of the operated system. These are a generic concepts and mechanisms that can be applied to many use cases and domains. This also means intent will be used and discussed in many different contexts. In order to facilitate a structured discussion about intent usage it is useful to classify intent.

A classification of intent was already done in the industry. This results in the frequent use of terminology that contextualizes intent. This is partly standardized or based on established best practices. This chapter structures commonly used classifications of intent. In this respect the intent user guide is not necessarily introducing new classes or types of intent, but tries to explain what a commonly used terminology means and in what contexts it is it useful.

In the TM Forum Autonomous Network the classification of intent will have no impact on the information models for expressing intent. The TM Forum Intent Ontology does only recognize one class of intent and does not allow the creation of subclasses for the purpose of establishing classification of intent in the information models. This is avoided, because typing and classification of intent is redundant information from the perspective of the autonomous system. For example, intent handlers on the service layer know already that they handle service intent. It is completely redundant information if the intent sent to this intent handler explicitly claims to be a service intent. All that matters is from the intent handler's perspective is that the intent expression contains requirements it understands and supports. This needs to be verified in detail at intent reception and the intent handler would need to conclude itself if the intent is valid and matches its autonomous domain. Explicit intent classification is therefore only relevant for human understanding of the use cases and scenarios, but not for the automated intent management system. If information about the use cases and context is helpful, it is always possible to add comments to the intent expression. These comments can then describe the use case and the intent content so that it is easier for human observers to understand the flow and procedures of the autonomous operation.

Categorization of intent can be done to address a variety of concerns. This means there are multiple dimensions of intent classification. This also means that an individual intent can be classified in different ways to describe various aspects of its content and use case.

Usually intent types are reflected by a classifying term in front of the word intent. For example, "Service Intent" or "Business Intent".

The following chapters structure commonly used intent categorizations and explain their meaning.

4.1. Categorization by source of intent

Intent can be distinguished by the source of the requirements expressed by the intent. This refers to the entity that has created an intent expression to communicate its requirements to an autonomous domain.

4.1.1. Customer Intent

A customer intent is an intent that originates from a customer. This typically refers to the customer of the CSP. The CSP is offering services to this customer. This intent is used by the customer to clarify its detailed requirements regarding what shall be delivered and how it is expected to perform.

4.1.2. User Intent

A user intent is an intent that originates from a service user. This typically refers to an individual user of a service offered by the CSP. This intent is used by the user to clarify its detailed requirements regarding what shall be delivered and how it is expected to perform. A user can also be a customer or the user can be an individual from a user group associated with an entity that has the customer role.

4.1.3. Operator Intent

An operator Intent is an intent that originates from entities that have a role within the network operator's network operation. This type of intent is therefore used by the network operator to communicate requirements to its own autonomous network. This term is independent of the layer of operation. It can therefore, for example, refer to intent used by business administrators to put requirements on autonomous business management processes. This intent would be an operator intent and also and business intent. Another example would be intent originating from the network operation technicians with the purpose to influence operational decisions and network behavior. For example, an intent can be used to inject additional requirements on security such as a minimum encryption level to be applied across all services.

4.2. Categorization by allocation of intent handling

Intent can be distinguished regarding the allocation of intent handling and targeted autonomous domain. This refers to the autonomous domain of the intent manager in the role of intent handler for an intent. The concept of autonomous domains was introduced by the autonomous networks reference architecture [ig1251-ra]. The reference architecture also allocates autonomous domains on the operational layers of business operation, service operation and resource operation. These concepts are the base for the intent categorization discussed in this chapter.

4.2.1. Business Intent

Business intent is an intent to be handled by an autonomous domain and intent handler allocated on business operation layer.

4.2.2. Service Intent

Service intent is an intent to be handled by an autonomous domain and intent handler allocated on service operation layer. Note that service intent can also mean intent associated with a service order as described in chapter 2.3. It depends on the context, what the term is meant to refer to. To avoid ambiguity it would be recommended to use more specific the terms such as "Service Layer Intent" if it shall refer to any intent that expresses requirements for service operation layer or the term "End-to-end Service Layer Intent" if end-to-end service layer shall be differentiated from resource domain specific services.

4.2.3. Resource Intent

Resource intent is an intent to be handled by an autonomous domain and intent handler allocated on resource operation layer. Resource intent can also refer to intents associated with services that abstract resources, because these Resource-Facing Services (RFS) are considered to be part of the resource operation layer.

Resource intent can be further distinguished by resource domain. Intent specific to these resource domains can be further classified. For example: RAN Intent, Transport Intent, Core (Network) Intent, Cloud Intent, ...

4.2.4. Operational Intent

Operational Intent is an intent that expresses requirements for network operation. Network operation refers to set of processes that implement certain aspects of network operation. Examples are service fulfillment and assurance, provisioning, deployment, optimization and healing. Therefore, operational intent is intent that requires certain outcomes to be achieved by these operations processes.

4.2.5. Network Function Intent

Network Function Intent is intent that expresses requirements to be handled by a network function. This would also refer to autonomous domains that are bound in scope to network functions.

4.3. Categorization by target object of requirements

Intent has a target. A target is the object or entity that needs to comply to the requirements expressed by the intent. This also means intent can be classified with respect to the target of its requirements. The general pattern is that the classifying work in front of Intent refers to the resource or type of resource. The following are examples of this pattern:

4.3.1. Service Intent

Intent can express detailed requirements associated with services. This intent is typically established when a service order is made and then express the requirements the resulting service instance shall comply to. In this context intent has a similar role as Service Level Objectives (SLO). This usage of the term service intent is not necessarily bound to the layers on the reference architecture. For example Customer facing services are typically ordered from the end-to-end service layer, while resource facing services or resource service are abstractions of resources and typically ordered from the resource operation layer. Unless the meaning of the term service intent is clear from context it is recommended to avoid ambiguity between this usage of the term service intent and the description provided in chapter 2.2.2. Using additional classification of the intent in question should help to resolve ambiguity. For example, CFS Intent and RFS Intent could be used.

4.3.2. Slice Intent

Slice intent is an intent that expresses requirements a slice needs to comply to.

4.3.3. Sub-Network Intent

A Sub-Network Intent is an intent that expresses requirements a subnetwork needs to comply to.

4.4. Categorization by nature of requirements

Intent specifies requirements and requirements can also be classified, for example, with respect to the concern the requirement is addressing. Intent classification can inherit the classification of the requirements it expresses. Intent typically expresses multiple different requirements. This means that many intents quality for multiple classifications.

4.4.1. Security Intent

Security Intent is an intent that expresses security requirements.

4.4.2. Performance Intent

Performance intent is an intent that expresses performance requirements. This is typically expressed by goals based on key performance indicators or any metric that is used to measure network, resource, or service performance. Latency and throughput are typical performance KPIs. Other examples are availability or power consumption.

4.4.3. User Experience Intent

A user experience intent is an intent that expresses requirements regarding user experience. These requirements are typically expressed by KPIs that correlate with user experience. For example end to end latency is a major factor in online gaming experience.

4.5. Categorization by use case

Intent is a generic mechanism for managing requirements. It can be used i the realization of a great variety of use cases. The use case the intent is embedded in can therefore also be used for classifying intent. The following subchapters present examples:

4.5.1. Energy Saving Intent

An Energy Saving intent is used to express requirements that help realizing energy saving in the operated network.

4.5.2. Optimization Intent

In some cases additional requirements are introduced to optimize the network operation with respect to additional concerns. This intent does not prescribe that optimization shall be done, because continuous optimization towards all intent requirements are already a basic concern of intent handling and its typical control loops. An intent handler does not need to be told when to optimize. But it might need additional requirements if it shall optimize towards additional objectives. This means an optimization intent shifts the point the intent handler considers to be optimal.

4.6. Categorization by expression

Intent can be expressed in many different ways. The TM Forum Intent Ontology is one proposed model for intent expression, but there are other ways to express and communicate the requirements. Intent can therefore be classified with respect to the way intent is expressed.

4.6.1. Natural Language Intent

A phrase in natural language can describe requirements including the requirement's target and context. This can be used as intent expression. It is mainly relevant on the human-machine interface.

4.6.2. Domain Specific Language Intent

A domain specific language is similar to natural language, but it has stricter rules what phrases and expressions are allowed or not and how certain concerns shall be expressed. Also, this can be the base for expression of intent.

4.6.3. Formal Intent or Formally Modeled Intent

Intent that is expressed using formally defined expression models can be referred to as formal intent. Intent that is expressed according to the TM Forum Intent Ontology (TIO) would follow a model that is based on the resource description framework (RDF). Other SDOs model intent in other modelling frameworks such as UML. These modeling frameworks create a complete syntactic and semantic definition resulting in ambiguity free Intent expressions. They also provide extensive software tools for implementing intent handling.

Formal Intent can also be further classified by the modeling framework it is used: RDF Intent, UML Intent. It can also be classified by the serialization format that encodes the intent expression. For example: Turtle Intent, JSON Intent, YAML Intent.

5. Intent-driven operation concepts

Intent is the “specification of all expectations including requirements, goals, and constraints given to a technical system”. This definition points at key guiding concepts and that ultimately resulted in architectural and modeling decisions within the TM Forum Autonomous Networks project. This chapter of the user guide explains these concepts and the TM Forum recommendations for intent-driven autonomous networks.

If we define the intents as expectations on entities states, then “intent” is not a brand-new concept, it has always existed, even in the days of human operated networks or automated operation through imperative user interfaces and APIs. A key difference is that, in the past, intent generation and implementation were performed by human beings. Humans operated the networks with the knowledge about customer needs, ordered services, available resources, etc. They also know context, such as available budget and business strategy of their organization helping them to arrive at the right decisions. Most of the requirements were not formally defined in a consolidated way. They were simply knowledge the operation teams have acquired through various means. The sources range from documents such as work orders, tickets and manuals that describe operational procedures, to instructions from supervisors and management as well as common sense and general understanding of good and bad system behavior based on training and experience. Human teams can then apply this diverse knowledge in their operational decisions and actions. In semi-automated systems, they would use this knowledge

In autonomous networks the main source for knowledge about operational goals and their context is intent. It is therefore necessary to separate the intent and from the system implemented to operate the network. Intent is knowledge available to the autonomous system. Intent is replacing many of the various sources of knowledge a human team has. This also means that intent needs to convey diverse knowledge that would allow operational accuracy and correctness that matches or even exceeds the abilities of human teams in network operation.

We distinguish an entity that defines knowledge about requirements, goals and diverse context using intent and the intelligent autonomous agent that implements the means for meeting these requirements. They interact through an intent interface. This requires a clear definition of intent, vocabulary and grammar for expressing intent and defined mechanisms to interact based on intent and manage their lifecycle as knowledge objects. This needs to be aligned with networks architecture with its layered abstraction of resources and services. This abstraction means that different knowledge expressed by intent wound be needed at different points in the networks architecture and different subsystems that implement network operation. All these concerns and implications are considered in TM Forum's recommendations for intent -driven autonomous networks and explained in the following sections of this user guide.

5.1. General Properties of Intent

5.1.1. Intent specifies entities that need to comply

Intent applies to objects and entities. This means that there is an entity or set of entities, which are controlled by an autonomous domain, that need to comply to intent expectations. In the TM Forum Intent Ontology (TIO) this entity is referred to as target. For example, the requirement shall be that the latency on a network slice shall be below 10ms. In this example the slice instance is the target that needs to comply with the requirement. The statement that the latency shall be below 10ms is a condition associated with the expectation that expresses requirements for this target. In the TIO models, targets are always associated with expectations.

5.1.2. Intent expresses goals about states

The way that targets and expectations are modeled by the TIO means that intent expressions define goals regarding the expected state of the autonomous domain.

5.1.3. Intent is declarative

Intent declares the wanted results rather than prescribing a specific solution. This means, intent would not prescribe or require an action to be executed, a configuration to be applied. It rather declares the expected results and leaves it to the autonomous network to find suitable actions and configurations.

5.1.4. Intent leaves room to find new solutions

The declarative and non-prescriptive nature of intent expression means that intent leaves room for the autonomous network to explore options while finding and, if needed, changing solutions. This means the declarative nature of intent is a direct enabler for the use of artificial intelligence (AI) in operational processes. AI can identify possible new solutions. As long as these new solutions meet the intent requirements they are viable. This level of autonomous of operation is different from pre-defined policies and rules that cannot diverge from the implemented solution strategies unless new strategies are implemented by the policy developer. This capability of AI to increase the level of autonomous behavior and self-optimization requires that the system has the authority to diverge and explore rather than apply pre-defined recipes. The declarative nature of intent leaves room for exploring and adapting.

5.1.5. Intent conveys preference

Ideally, intent expresses further knowledge beyond the declarative requirements. This establishes knowledge regarding preferences and value of outcomes. Examples are priorities between requirements and utility of reached states. This helps optimize autonomous decision-making and resolving conflicts by selecting preferential and therefore the best solutions out of all strategies and action available to the intent manager.

5.1.6. Intent is infrastructure agnostic

The expectation expressed by intent originally originates from contracts and business strategy. Those do not change or consider if the underlying system is replaced or modified. They typically do not mandate or imply a certain network topology or technology as long as the network can deliver the agreed services. This is achieved by layered abstraction of the infrastructure. Intent is expressed using the same abstractions. It is therefore to great extent infrastructure independent and agnostic.

5.1.7. Intent is portable

Differences between networks with respect to their capabilities, topology and technology will continue to exist. Also, implementations by system vendors will continue to differ in detail. The layered abstraction used in network operation and agreed on through standardization ensure that intent can be expressed in a way that is understandable across implementations and generations of networks. This means intents are portable in the sense that the same intent expression can be understood by a variety of systems that operating aspects of the network. They way these systems would meet the intent expectations can vary considerably, but as long as they understand the same intent and deliver compliance to its requirements the intent is fully portable.

5.1.8. Intent establishes strong separation of concerns

Intent is based on declarative abstraction of requirements rather than imperative ordering of actions. This strengthens the separation of concerns between the autonomous domains involved. With imperative actions, the requester needs to implement some understanding of what the action would do within the receiving autonomous domain. This often requires understanding of internal details of that domain. For example, the managed resources and their configuration might need to be shared. With intent based interactions detailed internal knowledge is not exposed anymore. The interaction is based on the requirements expressed in intent and if the receiving autonomous domain is compliant to them. How the receiving autonomous domain configures its resources and what actions it is taking is primarily a domain-internal concern.

5.1.9. Intent is complete

The requirements an autonomous system has to follow can be introduced in two fundamental ways. Requirements can be considered in the implementation of the management system and its policy rules. This means an autonomous domain would always consider these requirements in all its decisions and actions. However, some requirements needs to be handled dynamically as they represent varying needs of the CSP and its customers. Intent allows us to set and manage requirements dynamically. Furthermore, intent completes the knowledge of an autonomous domain regarding what requirements it needs to comply to. In this respect intent is complete. On top of the behaviors implied by the systems implementations due to its offline design requirements, intent is setting all additional requirements to be considered. Beyond that, the system has freedom to explore. This also means that a concern that is not managed through intent and is not already covered by the system implementation, including its policies, would not need to be considered. Consequently, intent based operation requires analysis and awareness if all relevant concerns are covered.

5.1.10. Intent is composable

A single intent can contain multiple requirements in the form of multiple expectations. And these multiple requirements can be concerned with different entities that need to meet the requirements. This means requirements for multiple concerns and use cases can be composed into a single intent and therefore life-cycle managed together as a set.

5.1.11. Intent is additive

An autonomous domain can receive intent from different sources. All received intent expectations are additive. The requirements an autonomous domain has to comply to the superset of all requirements across all received intents. This also means that managing the lifecycle and content of intent practically means managing partial sets of requirements.

5.1.12. Intent has an actively managed life-cycle

Intent requirements need to be actively met by an autonomous domain, while they are present and valid. This means that adding intent would add requirements which the receiving autonomous domain needs to meet until the intent is taken away. Intent therefore determines fulfillment as well as assurance of these requirements. The autonomous domain can stop assuring these requirements only after they were removed, be removing the intent. If the requirements are not needed any more the autonomous domain can stop monitoring it. If intent is removed, the autonomous that has handled the intent might conclude that it is not in an optimal state any longer, because it is configured and uses resources for requirements that are no longer applicable. It can take actions to change that and find a state that is preferential for the remaining requirements.

For example, an intent that specifies requirements for a network slice would not just determine how the slice is initially provisioned. Intent is rather the reason for monitoring the slice and assure the continued compliance to the requirements.

5.1.13. Intent is measurable

Intent specifies requirements in the form of expectations and conditions. Typically, these requirements state goals and targets based on system state, which involves metrics and KPIs. The autonomous domain that has received an intent using certain metrics is obliged to continuously assure compliance to the stated requirements. This means the autonomous domain needs to measure the metrics involved for observing the system state. An intent can therefore only use metrics in the intent expression that can be observed by the receiving autonomous domain. This is one of the most important limitations when planning network operation with intent. A use case that expresses requirements that cannot be measured by the network is a use case that cannot be implemented. Evolving the measurement and analytics infrastructure to meet the need of planned use cases is therefore an essential activity in the introduction of autonomous network operation. Intent asks the autonomous network to control certain aspects of the operation, but it can only control what it can measure.

5.1.14. Intent is comprehensible

Intent must be understandable by humans while being formally and unambiguously specified allowing it to be processed by machines. Reading the intent should allow understanding what is required. It must be comprehensible in the context of autonomous domain it specifies requirements for. The TIO is based on modelling techniques that have textual representations that allow human understanding of the intent expression with low requirements on specialized competence. For audiences without technical training it is recommended to implement intuitive presentation interfaces that further reduce barriers in understanding the intent.

Ideally the decisions of the autonomous domain and the resulting actions should also be explainable from the intent requirements. However, the volume of tasks and actions within an autonomous network and interdependencies between intents might require software support to present and explain the system behavior in a comprehensible way. The use of AI-based solutions might require to apply AI explainability techniques. This is outside the scope of the autonomous networks project and this user guide.

5.1.15. Intent is not ambiguous

Intent is used for the communication of requirements. This means that both parties involved and any observing party must arrive at the same understanding of what is required and what is achieved while trying to meet the requirements. Formally defined intent models such as the TM Forum Intent Ontology (TIO) are designed to remove ambiguity. The model defines vocabulary ti be used and its meaning. For other intent that does not rely on formally defined models for intent expression ambiguity can be a problem. This is in particular the case for intent expressions based on natural language, but that is outside the scope of the TIO.

5.1.16. Intent separates the requirements from operation

Intent is not doing or executing something. It is rather setting a requirement that something shall be achieved. There is a clear separation between the information object called intent, the entity that defines requirements and the autonomous network or an autonomous domain within that needs to comply to these requirements. The requirements are therefore separated from the actions used to comply to them. In TM Forum autonomous networks the function that defines requirements and then manages their life-cycle is referred to as "intent owner". The entity that receives intent is obliged to take action to comply to them is referred to as "intent handler". Intent is a knowledge object exchanged between an intent owner and an intent handler over an intent interface.

5.2. Intent in the Autonomous Networks Architecture

The reference architecture for autonomous networks as proposed by the TM Forum Autonomous Networks Project (ANP) is a functional architecture that provides structure to the functional components involved in autonomous network operation. It defines layers of operation, autonomous domains as subdivisions of an autonomous network and reference points for interfaces between functions and domains [ig1251-ra].

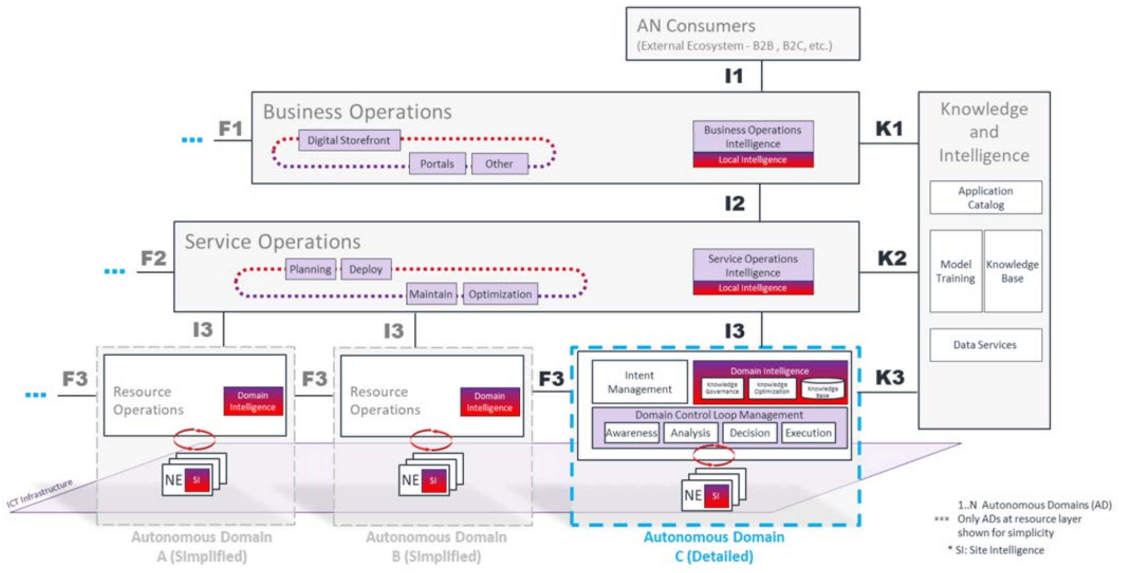


Figure 5-1. Autonomous Networks Reference Architecture [ig1251-ra]

The reference architecture considers that functions needed for operation and the intelligence they implement is decentralized and distributed. They are localized within domains referred to as Autonomous Domains. Autonomous domains are logical units of management. Complexity is encapsulated with these domains to compartmentalize capabilities. An Autonomous network can be assembled from multiple federated autonomous domains horizontally and vertically. It is therefore the building block of autonomous networks and autonomous operation.

The reference architecture distinguishes three layers of operation:

Business Operation is concerned with customer interactions with respect to contracts, SLAs and products.

Service Operation is concerned with end-to-end realization of services

Resource Operation is concerned with managing the network infrastructure. This includes resource utilization and configuration of network elements.

Each layer can be subdivided into autonomous domains with distinct scopes of responsibility for the realization of an autonomous network. These layers can further be subdivided. For example, in resource operation it is common practice to distinguish resource management from network functions.

Autonomous domains interact via reference points through open APIs. Autonomous domains within an autonomous network are recommended to be intent aware. This means, they can communicate by setting requirements through intent. Please note that intent capabilities will evolve over time and can be introduced successively into autonomous domains. Therefore, hybrid autonomous networks that are only partly intent aware will be reality for the foreseeable future. These hybrid autonomous networks consist of autonomous domains that follow the intent paradigm of managing declarative requirements, while others are not and primarily use conventional interfaces. We expect a gradual introduction towards intent based operation fueled by business cases that benefit from raising the level of autonomy by utilizing intent management capabilities.

The "I" and "F" reference points can use intent interfaces. TM Forum open APIs have introduced intent in multiple APIs. For example TMF921 is a dedicated API for intent management. If intent is used as part of product and service management, the respective APIs also include intent capabilities. This includes TMF641 Service Order, TMF622 Product Order, TMF645 Service Qualification and TMF679 Product Qualification.

5.3. Intent management Function

An intent-aware autonomous domain interacts with other autonomous domains by exchanging intent. This means that an autonomous domain needs to implement the respective management of and operation with intent based expectations. This function is referred to as "Intent Management Function" (IMF). An instance of an intent management function is referred to as "intent manager".

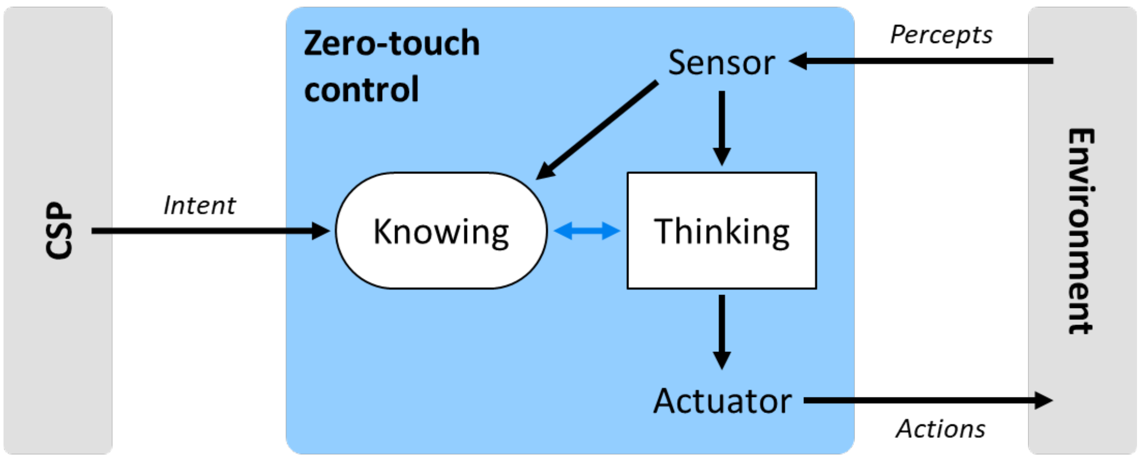


Figure 5-2. Intent Management Function implementing intelligent agents and zero-touch control

Intent management functions can be considered to be intelligent agents following the definition of intelligent agents in [ai-modern]. Used as base for autonomous networks these agents are used to realize the zero-touch paradigm of autonomous operation. Figure 5.2 shows an intelligent agent with the addition of intent as source for knowledge.

Intent Management Functions interact with the environment. In autonomous networks this environment is the autonomous domain and the resources and entities that are included in the scope of the autonomous domain. The Intent manager observes the state of the environment. This constitutes knowledge. Intent is in this respect additional knowledge added by the CSP and based on business needs. This means an intent manager has knowledge about the environment it controls and the requirements this environment needs to comply to. The central task of the intent manager is to find and execute solution strategies that would transition the state of the environment into compliance to the intent. Respective actions towards the environment will lead to a state change that can be verified through successive observations. This closes a control loop.

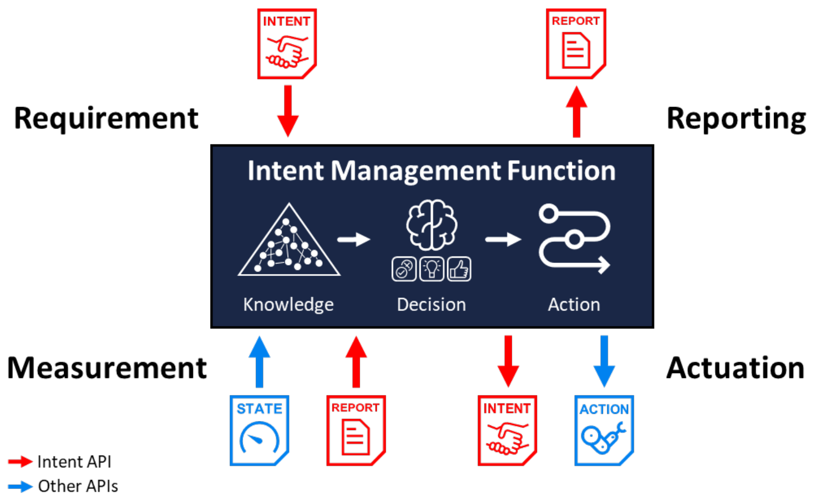


Figure 5-3. Intent management function

An intent management function receives requirements expressed by intent over an intent interface/API. It also utilizes the measurement and analytics infrastructure to observe the state and behavior of the autonomous domain. Receiving intent and receiving measurements creates knowledge that indicates which requirements the autonomous domain complies to and which does not. If there are requirements that are not met, this indicates that an analysis of the situation corrective actions to heal and optimize are being needed. The intent management function implements a decision process that identifies possible solution strategies and ultimately decides which strategy to apply and how to act. Depending on the nature and scope of the autonomous domain, solutions and actions can vary. They can, for example, involve changes in the configuration of network functions, ordering and provisioning of services or deployment of resources. The chosen solution will then be implemented through suitable actions towards the network. Any available management interface can be used depending on the nature of actions and managed entities to apply the action to.

If the solution chosen by an intent manager involves other autonomous domains need to behave in a particular way a suitable action would be to send intent into that domain with requirements that reflect the needed solution. This means that an intent manager meet the requirements it handles and need to comply to by defining a new set of requirements it needs other domains to comply to. It is doing a breakdown of requirements. This process is often referred to as "intent translation". It starts with received intent requirements that express terminal goals of the intent manager. It needs to find a way to comply to these requirements. It is its responsibility to do so. This intent manager might conclude that, in order to comply to its requirements it needs other autonomous domains to support is efforts by complying to a distinct set of requirements. These requirements constitute instrumental goals. The intent manager needs others to comply to them for the sake of reaching compliance to its own requirements.

Actions taken by the intent manager will lead to changes in the behavior of the autonomous domain and this is reflected by new measurements and observations that indicate compliance to the intent or not. This closes a control loop within the intent management function.

Intent managers are also obliged to provide intent reports. This means an intent manager that handles the intent and tries to meet the requirements would need to keep the originator of the intent requirements informed about progress and success. Sending and updating intent in one direction and reporting back establishes another control loop between two intent managers. This control loop is enabled by the intent interface/API.

5.4. Intent managers within autonomous domains

Within an intent based autonomous network intent management functions are implemented by autonomous domains. This means a multi-layer autonomous network is also a multi-layer architecture of diverse instances of intent management functions interacting with each other through an intent management enabled API.

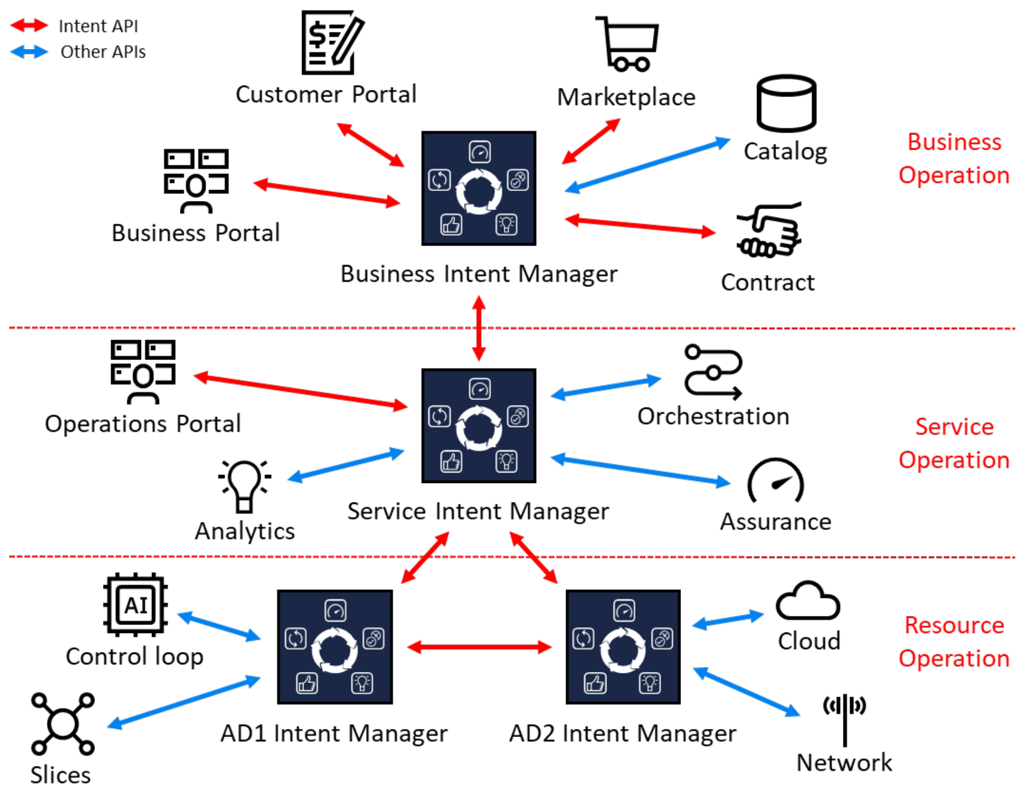


Figure 5-4. Architecture consisting of intent managers

Figure 5.4 shows an example of various intent managers allocated throughout the layers of an autonomous network. It is common practice to refer to intent managers using the level and autonomous domain. For example, a business intent manager is an instance of an intent management function that implements the handling of intent received on the business operation layer. A RAN intent manager would then be an instance of an intent management function allocated in resource operation layer and specifically concerned with the autonomous domain of RAN management.

Each distinct autonomous domain that is intent aware implements an intent management function. Also, each instance of an intent management function is associated with one autonomous domain. The intent management function implements the intent based interaction with other autonomous domains, and it implements intent fulfillment and assurance within its domain. This typically involves other functions of the domain. For example an intent manager in the service operation layer would involve service orchestration and assurance. In this respect an intent management function is an abstract function. It is the set of all functions within an autonomous domain that are intent aware and participate in intent fulfillment and assurance. The implementation of intent managers would be specific to the scope and responsibilities of its autonomous domain and can be significantly different. However, there are a few common properties of intent managers:

Intent managers implement an intent API to exchange intent and intent reports with other intent managers.

Intent managers monitor the compliance to intent requirements

Intent managers implement intent fulfillment for achieving compliance to intent requirements.

Intent managers implement an intent assurance control loop. In this loop they monitor compliance to intent, decide on operation strategies, and act to achieve or preserve compliance.

5.5. Roles of intent managers

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

Intent Owner
The intent owner is the origin intent requirements. It has created the intent expression to communicate its requirements to other intent managers. The intent owner is also responsible for managing the lifecycle of its requirements. This includes modifying the intent if needed and finally removing the intent object if it does not need compliance to these requirements any longer. Only the intent manager in the role of an intent owner is allowed to create, modify or remove intent.

Intent Handler
The intent handler receives the requirements it needs to comply to from an intent owner, and it operates the autonomous domain it is responsible for accordingly. Intent handlers do not modify or delete intent, but they can initially reject it. However, once accepted they are obliged to comply to the intent requirements as well as possible according to the means available to them. Intent handlers report their success and progress regarding an intent to the intent owner using intent reports.

Intent owner and intent handler are roles. By convention and common practice these terms are also used refer to an instance of an intent management function that assumes the respective role for a given intent. Therefore, the term "intent owner" is used to refer to an instance of an intent management function that has assumed the role of intent owner for an intent. Similarly, the term "intent handler" refers to an instance of an intent management function that assumes the role of intent handler for an intent.

An intent owner can express multiple distinct intents and send them to multiple intent handlers. Furthermore, an intent handler can receive multiple distinct intents from multiple intent owners. However, every individual intent has exactly one owner and one handler. An intent therefore establishes a unique relationship between two intent managers with clearly defined roles. Nevertheless, an intent manager can be the handler of some intent and at the same time the owner of other intent. This typically happens if the actions found in intent handling involve sending further requirements expressed within new intent to other intent managers. An intent handler can therefore act by sending intent and therefore become an owner of the new intent.

5.6. Intent life-cycle

Intent managers participate in the life-cycle of intent according to their roles. The autonomous networks project defines an intent life-cycle consisting of 5 phases. Please note that this is the live-cycle of intent as managed through the intent API and in collaboration of intent handlers and owners. This is different from the intent handling state machine that is used within an intent handler to keep track of its intent based operation.

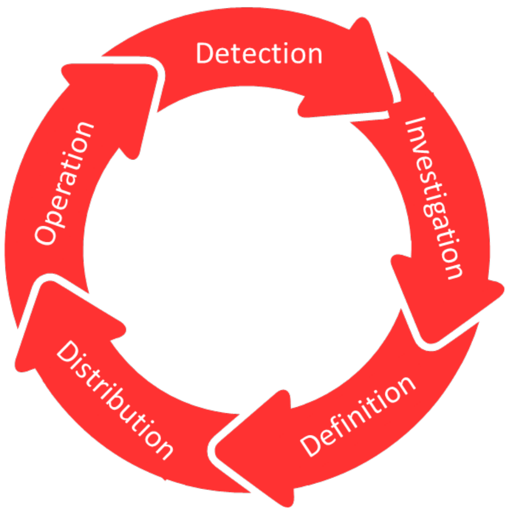


Figure 5-5. Intent lifecycle phases

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.

5.7. Intent control loops

Intent managers are involved in three different types of control loops shown in Figure 5.6.

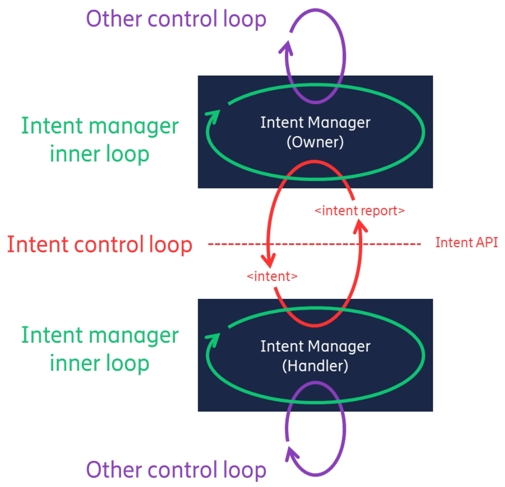


Figure 5-6. Intent control loops

Intent control loop - intent life-cycle management loop : This is the control loop related to the intent life-cycle management. It is executed between two intent managers in distinct roles of intent owner and intent handler for the intent. This control loop is executed over an API with intent management capabilities. 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 assurance loop: Intent managers have to continuously monitor and verify that the autonomous domain complies to the intent requirements and take healing and optimization actions. This is the intent assurance loop assurance executed within the intent handler. The subject to be assured are requirements expressed by the intent. This intent assurance loop interacts with the intent life-cycle management loop.

Other control loops: Intent managers operate in a network environment that is typically not fully adapted to intent based operation. There are control loops that are not intent aware. The intent manager can participate in these loops.

An intent owner might derive requirements from its role as participant in a control loop that is not intent driven. In this scenario the intent owner generates intent and sends it to an intent handler for defining and influencing the behavior of that autonomous domain as needed for the non-intent aware loop.

Another possibility is that intent handlers act by delegating control tasks to another control loop that is not intent aware. It would act through the interfaces exposed by loop control functions. This might involve deploying new loops or configuring existing loops. The loop is configured to comply to the intent requirements of this intent handler. This employment of and interaction with a control loop is shown in Figure 5.7.

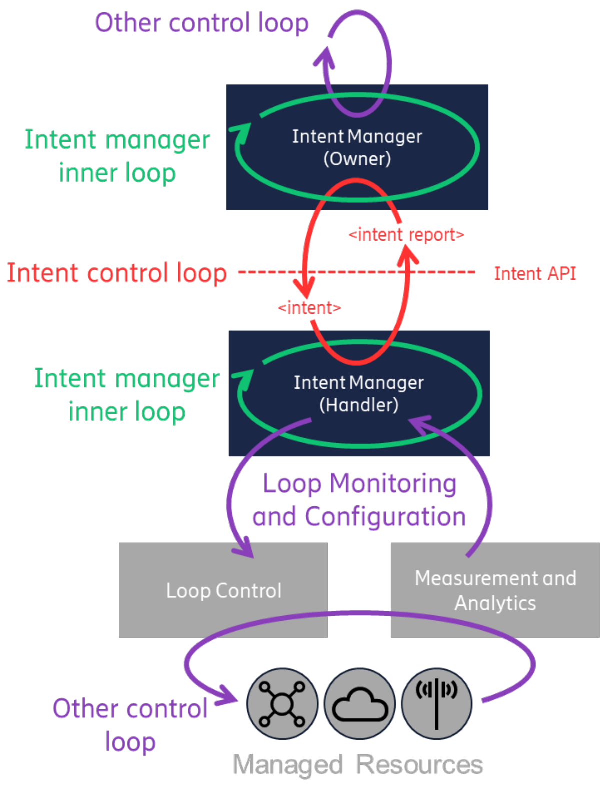


Figure 5-7. Interaction of intent handlers with other control loops

A typical scenario for this interaction between intent handing and control loops are control tasks with real time requirements. Intent handlers are typically not optimized to meet real time operation requirements. Intent handling typically involves many distinct tasks 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 need to be executed. Many of the tasks in intent handling involve evaluation capabilities or even complex predictive models for assessment of risk and mitigation of conflict. Complex processes like this can usually not be executed, while also meeting challenging real time requirements. The reaction intent handlers might still be fast compared to human operation, but still on the level of best effort and without the capability to deliver controlled or guaranteed response times as needed in real-time control. Low latency control loops with controlled real-time characteristics typically require loop processes optimized for efficiency with defined execution time budgets. Extensive evaluations and exploration of alternatives are usually not possible in real-time optimized loops. This means that demanding real-time requirements inhibit the direct use of intent in the control loop logic.

Intent handlers and specialized control loops can however complement each other. The intent handler would manage intent and monitor the results achieved by the real time control loop. Domain specific measurement and analytics systems provide the needed observations. The intent handler can therefore validate if the control loop's actions towards the networks nodes and resources are in line with the intent requirements and reach compliance. If not, the intent handler can process new loop configurations that are expected to result in better real time control performance compared to the intent that is currently applicable. This also means that changes in the intent might change the evaluation of the real time loop performance. The intent handler would act with loop configuration changes to continuously adapt what the loop is trying to achieve. This way, the real-time enabled control loop is indirectly intent aware and can adapt to new or changing intent. The real time loop controls the resources, while the intent handler controls what the real-time loop is trying to achieve.

6. Introduction to Ontology Models

6.1. Information modeling for intelligent autonomous networks

An autonomous network is an intelligent system with a scope of use cases that originate from network operation. It is an intelligent system that is expected to make reliably good decisions while facing complex dependencies. This requires technologies that allow implementing decision-making by a machine, thus creating artificial intelligence (AI). It addresses all kinds of decision problems, involving, for example classification, evaluation, prediction, planning and all kinds of concerns that were typically handled by human intelligence in traditional network operation. Two main approaches to creating intelligent systems were developed in AI research:

Numeric methods, such as machine learning, are typically based on statistical algorithms. Models are created, which numerically represent the environment and in particular the concern to be addressed within this environment. The environment is represented by its observable and typically numerical properties, which are relevant for the use-cases to be addressed. The training of the models tries to minimize the discrepancy between the model's representation of the environment or concern and observed reality. When these models are used for inferencing, they allow making similar decisions and produce similar results as presented to them in training.

Symbolic methods create models that abstract the components and concerns of the environment. Identifiers are assigned to represent relevant components of the environment and relevant concepts. This creates a symbolic vocabulary that is then used to establish knowledge about the relationships between the components. Machine reasoning methods such as first order logic can then be used to draw conclusions that are logically proven by the knowledge represented in the model.

Ontology graphs as used in intent modeling at TM Forum are a symbolic representation of knowledge. They are rooted in symbolic AI methods. A successful implementation of autonomous networks will require a combined use of various AI techniques. Symbolic ontology graphs were chosen for intent modeling due to its ability to handle the abstract and diverse nature of requirements. Ontology graphs allow modeling and therefore communicating all relevant semantic context needed by a machine for understanding intent. The model and its semantics are available in the online system and are combined with the system's observed state. This means the operated system or networks can be dynamically associated with the modeled semantics. This allows us to use machine reasoning methods to draw adaptive conclusion from the model. Autonomous networks at higher autonomy levels is expected to require a combined use of various artificial intelligence techniques. The choice of an information modeling framework that is directly rooted in AI tooling was therefore chosen to facilitate respective implementations.

TM Forum has chosen an ontology and symbolic AI approach over other information modeling approaches to facilitate symbolic AI and machine reasoning methods to be available. Other information modelling approaches are typically used as input in the system design and implementation process. A common example for this type of modelling is the Universal Modeling Language (UML) [uml2]. The models are typically used as input and artifacts in the software design process. Often the model is mainly concerned with schema for information and data, while most of the meaning of the information objects is described in the documentation rather than captured by the model. This information targets the development team. A software designer or data scientist would interpret the model in the context of use cases. This results in a software implementation that can automate these use cases and involve input and output information according to the model and its data schema. However, the model the development is based on is replaced by a more or less flexible program. An ontology-based approach contains all information that would also be in other information models. It can therefore also deliver input to software components similarly to information models. However, an ontology would preserve the entire semantic context of the information elements in the online system. This allows implementations that dynamically can re-iterate their behavior solely based on inference from the model. Machine reasoning tooling is available. This addresses an important aspect of autonomous networks: Adapting behavior to new situations and demands without human intervention, which includes the need to change the implementation.

The following chapters explain the modeling language and framework chosen for intent modeling. This is the foundation for the TM Forum Intent Ontology (TIO). This description will also explain how to read the models presented in the TM Forum specifications and help to understand the models and provided examples.

6.2. The Resource Description Framework

The TM Forum Intent Ontology is modeled using the Resource Description Framework (RDF). RDF refers to a standard by the World Wide Web Consortium (W3C).

RDF is intended for situations in which information on the Web needs to be processed by applications, rather than being only displayed to people. RDF provides a common framework for expressing this information, so it can be exchanged between applications without loss of meaning. [rdf11-primer]. The RDF data model provides a way to make statements about resources. It is a general framework for representing interconnected data and use this representation in data interchange. It was originally created as the base of data representation in the semantic web. RDF allows organizing information based on its meaning. This capability made it a common choice for implementing knowledge management use cases also beyond semantic web.

RDF extends the linking structure of the Web to use IRI/URIs [rfc3986][rfc3987] to name the relationship between things as well as the two ends of the link. This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual explanations. This triple of two graph nodes connected with a directed edge form an RDF statement. Using this simple model, RDF allows structured and semi-structured data to be mixed, exposed and shared across different applications. Collections of related RDF statements comprise a directed graph that maps the relationships among entities.

RDF statements express relationships between resources and thus establish information and knowledge about them. RDF does not limit the scope of what might be a resource; rather, the term "resource" is used in a general sense for whatever might be identified by an IRI. This includes documents, physical objects, people, abstract concepts, data objects, etc. In principle anything within an RDF model is a resource. This also means that the use of the term resources in the context of RDF models must not be confused with other definitions and usage of the term "resource" within specific application domains and contexts.

A collection of RDF statements about related entities can be used to construct an RDF graph that shows how those entities are related. This is a very flexible concept that allows modeling of all kinds of relationships, for example: A taxonomy classifies resources as a type of other resources. A meronomy expresses part of relationships. It can state that a resource is included in or part of another resource. An ontology is in this respect a model that can express taxonomies, meronomies and all kinds of other types of relationships. In case of the TM Forum Intent Ontology (TIO), this flexible expressiveness of RDF is used to create vocabulary for describing requirements as part of an intent entity.

RDF has features that facilitate data merging even if the underlying schemas differ. It specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. This is particularly useful for integrating data originating from different sources. It serves as the base of model federation without central governance proposed for the TM Forum intent models. This made TM Forum intent models highly modular and extensible. The concept of model federation is described in further detail in xxx.

The World Wide Web Consortium (W3C) maintains the standards for RDF, including the foundational concepts, semantics and specifications for different formats. The first syntax defined for RDF was based on the Extensible Markup Language (XML). Other syntaxes are now more commonly used, including Terse RDF Triple Language TURTLE [rdf11-turtle] [rdf12-turtle] and JavaScript Object Notation for Linked Data (JSON-LD) [json-ld].

RDF is continuously developed and RDF 1.2 is already described in W3C Working Draft documents [rdf12-concepts]. However, for the time being the TM Forum intent ontology is based in RDF 1.1 [rdf11-concepts]. The changes in RDF will be monitored and if applicable the TIO will be upgraded to the latest version.

6.3. The RDF modeling stack

The Resource Description Framework (RDF) is a framework for representing information in the Web [rdf]. W3C has specified a set of models to be used in semantic web. The TM Forum intent ontology is build on this modeling foundation.

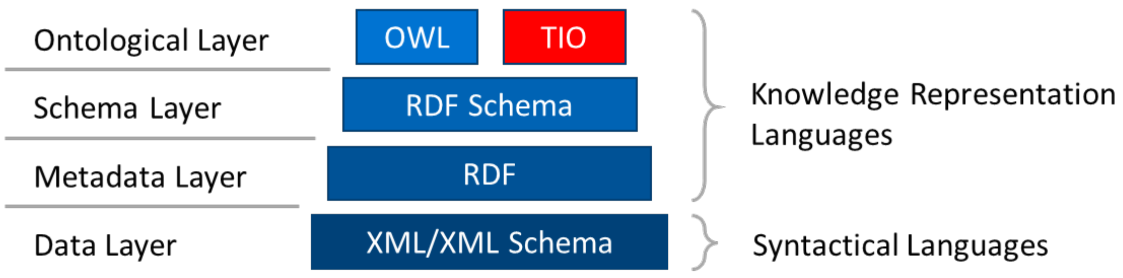


Figure 6-1. The RDF modeling stack

The Resource Description Framework (RDF) [rdf11-primer] defines a metadata layer. It establishes RDF graphs as fundamental modeling concept. RDF graphs are defined through RDF triples. Furthermore, they use of Internationalized Resource Identifiers (IRI) [iri] for identification of entities in the model. Furthermore, it establishes literals and datatypes rooted in XML Schema datatypes.

The RDF data model provides a way to make statements about resources. In practice, RDF is typically used in combination with vocabularies or other conventions that provide semantic information about these resources. To support the definition of vocabularies RDF provides the RDF Schema language (RDFS) [rdf11-schema]. RDF Schema provides a data-modelling vocabulary for RDF data. RDF Schema is an extension of the basic RDF vocabulary. Key definitions by RDFS are the Resource, Class, Literal and Datatype. Furthermore, it allows statements about the domain and range of properties. It also introduces containers and collections. This provides modeling vocabulary as base for further layers of modeling.

The ontological layer builds on RDF and RDFS. Ontologies are formalized vocabularies of terms, often covering a specific domain and shared by a community of users. They specify the definitions of terms by describing their relationships with other terms in the ontology. A frequently used modelling language on the ontological layer is the Web Ontology Language (OWL) [owl2-overview]. OWL provides extensive expressiveness. However, it bears challenges regarding computational complexity, completeness and decidability in reasoning implementations especially when handling large number of classes or instances with respect to reasoning and query performance. The OWL standard mitigates this by introducing vocabulary subsets, referred to as OWL profile. A profile addresses certain use case properties such as handling of large number of instance in the model.

As many of the advanced concepts in OWL are not needed in intent modeling, but can lead to high complexity in implementations, the TM Forum Intent Ontology (TIO) is not using OWL as base. It is rather directly based on RDF/RDFS as additional modeling language on the ontological layer.

The ontological layer can be further subdivided by models that build on each other for extending the vocabulary. This is often done to address domain and use case specific requirements. For example the Time Ontology in OWL [owltime] specifies vocabulary for expression of time concepts. It is based on and expanding the vocabulary of OWL. Another example is the TM Forum Intent Ontology. It consists of a general purpose intent modeling ontology as base for the intent common model and intent extension models.

6.4. RDF graphs and statements with triples

The core structure of the abstract syntax of RDF is a set of triples. Each triple each consists of a subject, a predicate and an object. A set of such triples is called an RDF graph. An RDF triple is conventionally written in the order of:

<subject>   <predicate>   <object>

Asserting an RDF triple says that some relationship, indicated by the predicate, holds between the resources denoted by the subject and object. This statement made and asserted by an RDF triple is known as an RDF statement. The predicate itself is a IRI and denotes a property, that is, a resource that can be thought of as a binary relation.

An RDF graph can be visualized as a node and directed-arc diagram, in which each triple is represented as a node-arc-node link. This means, subjects and objects are nodes in the RDF graph and predicates are represented by arrows that originate at the subject node and point to the object.

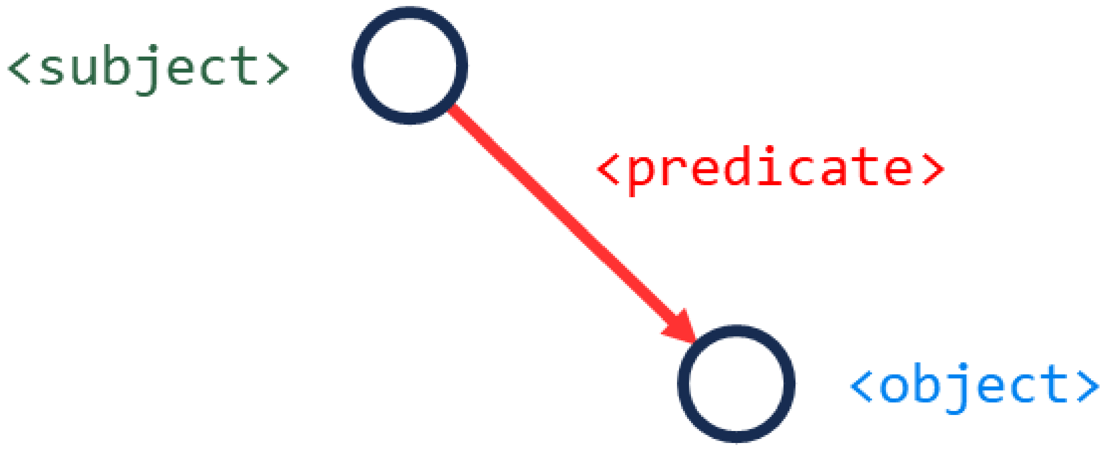


Figure 6-2. Graph representation of triples

Subjects are IRI or Blank Nodes. Predicates are IRI. Objects are IRI, Blank Nodes or Literals. More information on IRI, Blank Nodes and Literals is provided in the following chapters.

Using multiple RDF statements allows the construction of complex graphs expressing different types of relationships between entities. For example, different properties can be used with the same subjects and objects to assert further aspects of their relationship. If the object of one statement are used as subject of another statement, graphs are concatenated. This can lead to indirect relationships between resources that are not stated directly but can be inferred. Machine reasoning processes do this type of inference.

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.

6.5. Internationalized Resource Identifiers for identification

Each subject, predicate or object within the RDF graph can be identified by an Internationalized Resource Identifier (IRI) [iri]. An IRI is a Unicode [unicode] string that conforms to the syntax defined in RFC 3987 [iri]. IRI are a generalization of Universal Resource Identifiers (URI) [uri] and permits a wider range of Unicode characters. Every absolute URI and URL is a IRI, but not every IRI is an URI.

An IRI denotes something in the world. These things are called resources. Anything can be a resource, including physical things, documents, abstract concepts, numbers and strings. The term "resource" as used in RDF is therefore synonymous with "entity". The Resource Description Framework is in this sense a framework for describing any entity in the world.

Through rigorous use of IRI/URI, models based on RDF/RDFS such as the TIO have globally unique references for everything they model. This globally unique identification of resources is also the foundation for combining modular vocabularies into federated models.

Subjects, predicates and objects of RDF statements can be IRI. In the statement this IRI represents the resource it notes. In this respect the statement is about resources. A subject-predicate-object statement in RDF using IRI 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://tio.models.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.

The TM Forum Intent Ontology allows the definition of connectors that establish proxy IRI for identification of entities within domains that use other schema for identification. For example, 3GPP uses Distinguished Names (DN) within its object management. In the TIO, the translation between other identification schemes and IRI is subject to the work regarding connectors [tr293-connector].

6.6. Literals

Literals are used to identify values such as strings, numbers, dates. A literal can be used as object in RDF statements. A literal in an RDF graph consists of two or three elements:

- A lexical form, being a Unicode [unicode] string.
- A datatype identified by an IRI that determines how the lexical form maps to a literal value.
- If the datatype IRI is a string a language tag [bcp47-language]. A literal is a language-tagged string if the third element is present.

Please note that simple literals without datatype or language tag are allowed. The interpretation of the lexical form as value might then be ambiguous leading to warnings. In practice the datatype and therefore the correct interpretation of the lexical form as value might be inferred from context.

Examples:

`"1"^^xsd:integer`

This literal is an integer according to the datatypes specified in XLM Schema.

`"true"^^xsd:boolean`

This is interpreted as boolean value true.

`"1"^^xsd:boolean`

The boolean datatype also has alternative lexical form for the same value true.

`"true"`

This literal in simple form without explicitly stated datatype can have multiple interpretations. It can be a string value or a boolean. This ambiguity might be dissolved depending on how and where the literal is used. For example, if it is used in an RDF statement where the predicate is defined with range of `xsd:boolean` the correct interpretation as boolean truth value can be inferred.

`"hello world"@en`

This literal is a string with a tag that indicates it contains English language.

`"10 ms"^^quan:quantity`

This literal is a quantity according to the quantity datatype specified in the TIO [tr292d-quan]. A quantity has a numeric value in combination with an optional unit.

6.7. RDF Blank Nodes

At times, it becomes necessary to be able to express information without the need to uniquely identify the used nodes within the RDF graph with an IRI. This type of node is called a blank node. It is identified by a blank node identifier rather than an IRI.

Blank node identifiers are local identifiers that are used in some concrete RDF syntaxes or RDF store implementations. They are always locally scoped to the file or RDF store. Furthermore, blank node identifiers are not persistent or portable. Blank node identifiers are not part of the RDF abstract syntax, but are entirely dependent on the concrete syntax or implementation. How blank nodes are used and how blank node identifiers are expressed in TURTLE notation of RDF graphs will be explained in the following chapters.

6.8. Notation formats for RDF statements

An RDF graph consists of abstract elements such as nodes and edges that interlink the nodes. If information in the format of RDF graphs shall be communicated from one system to another via an API or stored in a document, a suitable notation format for expressing the graph is needed. For RDF there are several text based representations of the RDF graph:

TURTLE: Terse RDF Triple Language [rdf11-turtle] [rdf12-turtle]

TURTLE is one of the most used notations for RDF graphs. It has a compact form and specifies many syntax simplifications that aid human understanding of the modeled information. This property makes it a popular choice as based for human driven model creation and model inspection. For this reason TURTLE is also used throughout the TM Forum Intent Ontology in specifications and examples.

JSON-LD: JavaScript Object Notation - Linked Data [json-ld]

JSON-LD is a serialization format for Linked Data based on JSON [json]. It is therefore important to distinguish between the syntax, which is defined by JSON and the data model which is an extension of the RDF data model. TM Forum APIs are JSON based and therefore API developer have already substantial experience working with JSON syntax. This means JSON-LD is a recommended alternative to TURTLE in the context of TM Forum open APIs.

RDF/XML: XML notation for RDF [rdf11-xml-syntax]

This is an XML based notation syntax for RDF Graphs.

YAML-LD [yaml-ld-report]

YAML-LD is a set of conventions built on top of YAML, which outlines how to serialize linked data as YAML based on JSON-LD syntax, semantics and APIs. The emergence of YAML as a more concise format for representing information previously serialized as JSON, including Linked Data, has led to the development of YAML-LD.

All presented notation formats for RDF graphs are fully compatible with each other as they all can be used for expressing the same RDF Graph. This also means that translation between them is always possible and tools are available that can automate the process. The choice of working with one or several of these formats when creating and communicating RDF graph mainly depends on personal preference. It is recommended to choose the formats that match the environment in which the RDF graphs are used. In the context of the TM Forum intent ontology we recommend using TURTLE due to its human friendly syntax and using JSON-LD in the context of automated processes involving TM Forum open APIs, which are already based on JSON. The following chapters explain the syntax of TURTLE, which is used in TIO specification documents.

6.9. 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 quick to learn and intuitively readable by human users. Fundamentally TURTLE consists of RDF statements. However, RDF syntax is very efficient and concise in expressing RDF statements with useful shortcuts for elements of RDF graphs.

6.9.1. Prefix definitions for substituting namespaces

An RDF statement using full IRI for identification of subject, predicate and object can be made in TURTLE. However, full IRI strings can be long and are therefore hard to read. Much of an IRI string is typically repetitive. It denotes the namespace of the identified resource and this part is often repeated across many RDF statements TURTLE allows the definition of a short namespace substitute called prefix. A prefix definition line starts with the keyword "@prefix" followed by the prefix name that ends with a colon character ":". This is followed by the IRI namespace string, which is substituted by the short prefix name. The following example use TURTLE notation defining namespace prefixes.

```
@prefix    : http://example.com/IntentIndividuals/ .
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns# .
@prefix icm: https://tio.models.tmforum.org/tio/3.6.0/IntentCommonModel/
.@prefix  ex: http://example.com/ .
```

This allows an alternative shortened expression of IRI based RDF statements. An example RDF statement with fill IRI may be:

```
http://example.com/IntentIndividuals/Intent1
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
https://tio.models.tmforum.org/tio/3.6.0/IntentCommonModel/Intent .
```

The domain `example.com` is reserved for general purpose use in examples. Namespaces with this domain are often used in examples provided throughout the TIO specifications. Typically, the prefix "`ex:`" is chosen to represent an `example.com` based namespace. In production environments it is expected to use IRI schema based on domains that are owned and controlled by parties such as a CSP or a system vendor instead.

The same statement can be made using the defined prefixes as substitute of the namespace part of IRI string:

```
:Intent1 rdf:type icm:Intent .
```

This notation is much more readable as it focuses on the significant name of the resource avoiding the long namespace part. Prefixes with empty name are also allowed. They only consist of the colon character. This is often used for the most frequently used namespace in a document as it provides the most compact notation.

The prefix notation of IRI is used in documents and textual representations of RDF Graphs. This is however only a compact notation of an IRI and does not change the identifier used for a resource. It is still identified by an IRI.

6.9.2. Conventions and shortcuts

A statement in TURTLE syntax is terminated with a full stop character ".".

The keyword "a" is substituting the predicate rdf:type as defined in RDF. Its full IRI is `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`. Therefore, the following three statements are equivalent:

```
:Intent1 http://www.w3.org/1999/02/22-rdf-syntax-ns#type icm:Intent
.:Intent1 rdf:type icm:Intent .:Intent1 a icm:Intent .
```

The keywords "`true`" and "`false`" can be used instead of "`TRUE`" and "FALSE" as values of literals of the `xsd:boolean` datatype defined in XML Schema. For example, the following statements are equivalent:

```
rdf:value rdf:value "TRUE"^^xsd:boolean .rdf:value rdf:value true .
```

Using the reserved keywords "`true`" and "`false`" implies the literal has the datatype `xsd:boolean`. A correct interpretation of this literal value does not require that the datatype is explicitly stated to disambiguate strings from boolean values.

Literals are expressed in TURTLE as described in chapter 6.5. This includes the "`@`" character to denote language tags and the "`^^`" characters to connect the literal string with its datatype.

6.9.3. Predicate lists

Often multiple statements are made with the same subject, but different predicates and objects, for example:

```
ex:Intent1 a icm:Intent .
ex:Intent1 rdfs:comment "example intent"@en .
ex:Intent1 ex:somePredicate ex:SomeObject .
```

In TURTLE this set of statements can be expressed using predicate list notation. In predicate lists a statement is terminated with a semicolon ";" instead of a full stop. This means that the following statement has the same subject, which does not need to be repeated. The predicate list ends with a full stop ".". The following set of statements are the same as above but expressed using a predicate list.

```
ex:Intent1 a icm:Intent ;
           rdfs:comment "example intent" ;
           ex:somePredicate ex:SomeObject .
```

This example also shows typical formatting within TURTLE text files with indentation used for further enhancing readability.

6.9.4. Object Lists

Often multiple statements are made with the same subjects and repeated predicates, but with different objects:

```
ex:C1 rdfs:member ex:R01 .
ex:C1 rdfs:member ex:R01 .
ex:C1 rdfs:member ex:R01 .
```

In TURTLE notation the same can be expressed using an object list. In object lists a statement is terminated with a comma ",". This means that the following statement has the same subject and predicate, which do not need to be repeated. This predicate list ends with a full stop ".". The statements shown above can therefore be expressed as.

```
ex:C1 rdfs:member ex:R01, ex:R02, ex:R03 .
```

This one line of TURTLE notation therefore expresses three distinct statements.

6.9.5. Blank nodes in TURTLE

TURTLE provides labeled and unlabeled blank nodes (b-nodes).

A labeled blank node is a resource identifier that uses the reserved namespace prefix "_:". Labeled blank nodes (b-node) are identified by the "_:" prefix followed by its label. This means, that within a local environment, such as a document that contains RDF statements in TURTLE notation, a labeled blank node can be used similarly to an IRI to identify resources within RDF statements. This means, that within its local environment, several statements can be made using the identifier of the labeled blank node to refer to the same resource.

```
ex:E1 icm:target _:x .
_:x rdf:type icm:Target .
_:x rdfs:member ex:Slice1 .
```

In this example a named blank node with label "_:x" is used across multiple statements as subject or object.

Unlabeled blank nodes are represented by square brackets "[ ]".

```
ex:E1 icm:target [ rdf:type icm:Target
;                    rdfs:member ex:Slice1 ]
```

This notation allows nested statements. The statement with the icm:target property has the entire blank node as object. Within the square brackets that represent the blank node, multiple statements can be made that have the blank node as common subject. Because they share a common subject, these statements use predicate list notation with the semicolon character as statement separator.

Labeled and unlabeled blank nodes still need identifiers that are assigned and managed by the underlying system that handles the RDF graphs. For example, a database that stores the graph and provides interfaces for queries and inference based on the graph. The explicit names given to l labeled blank nodes are local to a set of RDF statements. An RDF graph would typically consist of multiple such sets of RDF statements forming multiple local naming environments. This means, that names chosen for labeled blank nodes can repeat within a graph and need to be managed accordingly with system generated transparent internal identifiers.

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, two distinct blank nodes are used. The first blank node is used as subject of a statement with the predicate foaf:knows. We know that this blank node represents a person as that can be inferred from being in the domain of foaf:knows. Other than that the first blank node has no properties, and we therefore do not know more about. This blank node can therefore be interpreted as "someone".

This example uses the friend of a friend (foaf) ontology. It provides vocabulary to express relationships between persons. A statement with the property foaf:knows expresses that the subject "knows" the object.

The second blank nodes in the object of the statement also represents a person. Here we know that this person has the property foaf:name with the literal value "Bob". So we know, whatever or whoever this blank node represents, has the name "Bob". So overall the statement expresses that there is a resource, which is a person that is in a foaf:knows relationship with another resource, which is another person that also has the name property with value Bob. Or in other words: There is someone who knows somebody else called Bob.

6.9.6. Containers

RDF defines containers to express sets of objects. Containers are instances of the class `rdfs:Container`. This class is a super-class of the RDF container classes rdf:Bag, rdf:Seq and rdf:Alt. Formally these container classes are not different, but they can be used to indicate to the user additional information. A container of class rdf:Bag is conventionally interpreted as unordered. Containers of class rdf:Seq can indicate that the order of elements is significant. Containers of class rdf:Alt can indicate that the elements are interpreted as alternatives with the first element as default choice.

The notion of order and preference in containers is derived from container membership properties `rdf:_1`, `rdf:_2`, `rdf:_3`, etc. These properties enumerate the container member elements. RDFS also defines the super-property `rdfs:member`. All container membership properties are sub properties of rdfs:member. Using these properties a container can be specified in multiple ways:

```
ex:SubscriptionTypes
  rdf:_1 ex:Gold ;
  rdf:_2 ex:Silver ;
  rdf:_3 ex:Bronze
.
ex:Cities
  rdf:member ex:Stockholm ;
  rdf:member ex:Uppsala ;
  rdf:member ex:Gothenburg ;
.
```

This example specifies two containers. Each of the containers has three elements. The example shows how the containers can be specified using statements with container membership properties or alternatively multiple statements with the rdf:member property.

In the TM Forum Intent Ontology it is recommended to use collections if order among the members of a set shall be expressed. The distinction of container subtypes is therefore not required. Instead, all containers in the TIO are of type `rdf:Container` without further distinction. Furthermore, because implied order is not used, the property `rdfs:member` is recommended to be used for enumerating container elements. In the TIO, the vocabulary regarding set operators [tr292f-set] depends on RDF containers as data structure for sets.

6.9.7. Collections

Collections are linked lists. They are used for expressing sets of objects in which objects are ordered. Each element of the list is represented by an instance of class `rdf:List`. The property `rdf:first` points to the object that shall be in the respective position of the list and the property `rdf:rest` points to the next element in the list. The list ends with no further elements by using to `rdf:rest` to point at an empty list represented by `rdf:nil`. The following example shows how a list with three elements can be expressed:

```
ex:L1
  rdf:first ex:O1 ;
  rdf:rest ex:L2
.
ex:L2
  rdf:first ex:O2;
  rdf:rest ex:L3;
.
ex:L3
  rdf:first ex:O3;
  rdf:rest rdf:nil
.
```

In this example the list `ex:L1` is associated with `ex:O1` as its first element, and it states that the list continues in `ex:L2`. The list `ex:L2` is associated with `ex:O2` and the list continues in `ex:L3`. The list `ex:L3` is associated with the `ex:O3` and it points at the empty list `rdf:nil` as next element to indicate that the list ends. This expresses that `ex:O1`, `ex:O2` and `ex:O3` are an ordered set. The graph representation of this collection is shown in Figure 6.3.

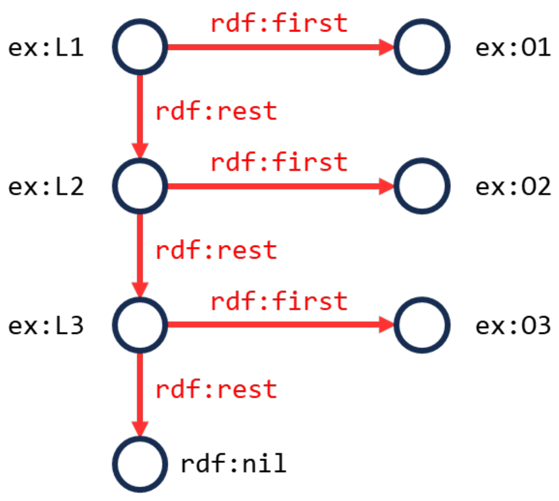


Figure 6-3. Graph of an example collection

Collections expressed directly using RDF statements are complex structures and therefore not easy to read. To make using collections easier, TURTLE provides a syntax for specifying them in much more compact way. In TURTLE a collection is represented by "`( )`" brackets. The list elements are then enumerated within the brackets. This means an equivalent list to the example defined earlier in this chapter and its entire internal structure and graph can be represented in TURTLE notation by "`( ex:O1 ex:O2 ex:O3 )`". The major difference between the notations is that list elements in the TURTLE notation are blank nodes, while they are identified by IRI in the full structured notation. However, in many practical use cases the main concern is to express the order of the listed objects rather than referring to the collection as a whole or its structural elements, thus blank nodes are sufficient.

Lists can be used in statements, for example:

```
ex:SomeObject1
  ex:preference ex:L1
.
ex:SomeObject1
  ex:preference ( ex:O1 ex:O2 ex:O3 )
.
ex:SomeObject1
  ex:preference [ rdf:first ex:O1 ;
                rdf:rest [ rdf:first ex:O2 ;
                           rdf:rest [ rdf:first ex:O3 ;
                                      rdf:rest rdf:nil
                                    ]
                         ]
               ]
.
```

This example contains three statements referring to collections in the statement's object. The first statement uses the first list element ex:L1 that marks the starting point of the example list as specified earlier in this chapter. The second statement uses the compact TURTLE notation to specify an equivalent list. The third statement uses the full RDF notation, but with black nodes as list elements. This is equivalent to the TURTLE notation as no IRI are used for the distinct nodes within the list structure.

Within the TIO, an important use case for collections are functions. Functions have argument lists. This means functions are properties with lists that represent function arguments in their range.

6.10. Property range and domain

The properties `rdfs:range` and `rdfs:domain` are used to establish knowledge about the subject and object in intent expressions using a predicate. In this respect they are properties of properties.

The property `rdfs:domain` asserts that the subject of a statement made with a property has a certain type. Similarly, the property `rdfs:range` states the same for the object of a statement made with a property. For example, in the intent common model, the property icm:deliveryType is described as

```
icm:deliveryType
  a rdf:Property ;
  rdfs:domain icm:DeliveryExpectation ;
  rdfs:range rdfs:Class
.
```

This assertion allows concluding that the subject of a statement with the property `icm:deliveryType` is `icm:DeliveryExpectation` and that the object of this statement is of type rdfs:Class.

It is important to note that domain and range are not meant to establish a rule about what types are allowed to be used and therefore shall be enforced. They rather allow inferring that whatever is used as subject or object in a statement are entities of a certain type.

6.11. TIO specific expressions

6.11.1. Function notation

The TIO introduces functions. A function is a property with a collection type object in its rage. This collection contains a list that is interpreted as function arguments. Furthermore, a function has a result. This result implies a statement with the `rdf:value` property in the function's subject.

For example, the `quan:smaller` function has two arguments of type `quan:Quantity`. Furthermore, it produces results of type `xsd:boolean`. It is used in the following example to determine the result of a condition:

```
ex:C1
  a log:Condition ;
  quan:smaller ( "100ms"^^quan:quantity ;
                [ rdf:value "1"^^xsd:decimal ;
                  quan:unit "s" ] )
.
```

In this example the function compares if the quantity of "`100 ms`" is smaller than the quantity "`1 s`". As this is true, the result value of the function assumes the value of "`true`". This is treated as substitute for a statement with `rdf:value` that has the function's subject `ex:C1` and the function's result value as object. This means the following statement is logically equivalent:

```
ex:C1
  a log:Condition ;
  rdf:value "true"^^xsd:boolean ;
.
```

Please note that this is en trivial example showcasing the principle of functions implying statements with `rdf:value`. The function result and therefore the equivalent value of `rdf:value` can be dynamically changing, depending on the nature of the function and the values of its arguments. The exact value is therefore always a momentary result of the function evaluation.

6.11.2. Value property in TIO

In many use cases one or several values needs to be associated with an entity in an RDF graph. RDF introduces the generic property rdf:value for this purpose.

In the TIO the rdf:value property is extensively used. For example, the value of a quantity object is expressed using rdf:value [tr292d-quan].

If a function property in TIO is used, it delivers a value to its subject. This result value of the function would be equivalent with an assertion made using `rdf:value`. For example:

```
ex:OB1
  a met:Observation
  met:observedMetric ex:latency ;
  rdf:value "100 ms"^^quan:quantity
.
ex:C1
  a log:Condition ;
  quan:smaller ( met:observedValue ( ex:OB1 )
                [ rdf:value "1000"^^xsd:decimal ;
                  quan:unit "ms" ] )
.
```

In this example the observation `ex:OB1` has the value "100 ms" explicitly asserted using the `rdf:value` property. A query to the knowledge base for the `rdf:value` of `ex:OB1` would therefore return the value "100 ms".

This example also demonstrates implicit value assignments as result of a function. The condition `ex:C1` uses the function `quan:smaller` to determine if the observed value represented by the observation `ex:OB1` is smaller than "1000". The `quan:smaller` function arguments are two quantities to compare. Here the first argument shall be the value of the observation `ex:OB1`. The result value of the function `met:observedValue` is the `rdf:value` of its argument. This shows how the result value of a function substitutes a `rdf:value` statement in its subject. When substituting in the result of the met:observedValue function, the equivalent quan spammer statement would be:

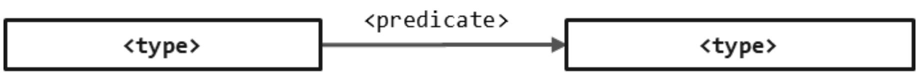
```
quan:smaller ( rdf:value "100 ms"^^quan:quantity
              [ rdf:value "1000"^^xsd:decimal ;
                quan:unit "ms" ] )
```

The result type of the `quan:smaller` function is boolean. This substitutes and therefore determines the `rdf:value` of the condition `ex:C1`. In this example the smaller condition it true. If a request to the knowledge base asks for the `rdf:value` of `ex:C1` the result would be the boolean value "`true`".

6.11.3. Statement diagrams

The specifications of the TM Forum intent ontology use statement diagrams to show how RDF statements can be constructed using the specified vocabulary. This shows how the vocabulary is intended to be used to build graphs from sets of statements. Statement diagrams are inspired by class diagrams, but they differ by providing custom expressiveness and additional features explained in this chapter.

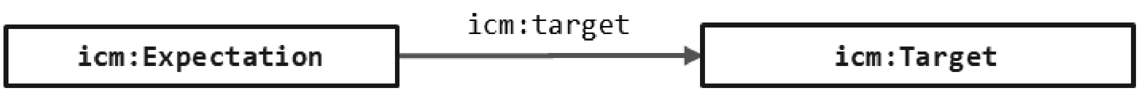
Statement diagrams consists of a few basic building blocks:



Rectangular boxes represent resources. Arrows represent predicates. In general this describes an RDF statement with the subject being represented by the box at the start of the arrow and the object being represented by the box at the tip of the arrow. The arrow represents the predicate of the statement. If multiple predicates are stated for a single arrow this means any of these predicates can be used in statements according to the diagram.

Statements with instances:

If a black solid arrow is used, the box is at the start of a arrow denotes the type of the subject in a statement with the predicate represented by the arrow. The box at the tip of the arrow denotes the type of the object of the statement. This means the start and tip of a black solid arrow are similar to domain and range properties of a predicate as they indicate the types of subject and object. For example:



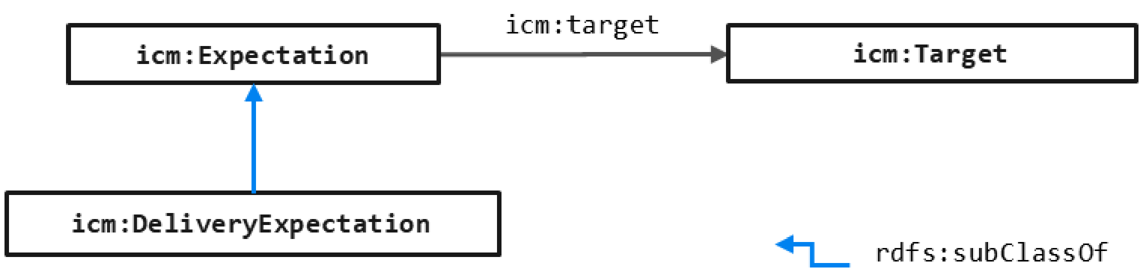
This example states that a valid statement can consist of an instance of class `icm:Expectation` as subject, the predicate `icm:target` and an instance of class `icm:Target` as object. The statement

```
ex:E1 icm:target ex:T1 .
```

is a valid statement according to the example diagram with `ex:E1` being an instance of `icm:Expectation` and `ex:T1` and instance of `icm:Target`.

Pre-defined arrows:

Other arrows can also be used In these diagrams the arrows can represent any predicate, which is usually stated with the arrow. Some diagrams can however use arrows that are colored or otherwise marked to represent a specific predicate. This is often used for predicates such as rdf:type or rdfs:subClassOf creating class diagrams. Typically, a legend explains the meaning of the specially formatted arrows. For example:

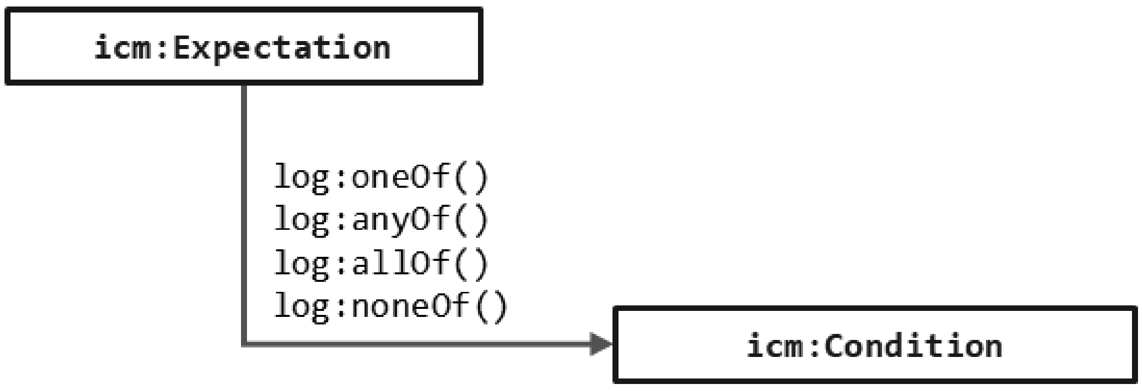


In this example the diagram states that `icm:DeliveryExpectation` is a subclass of icm:Expectation.

Function predicates:

The TIO defines functions. Functions are special predicates with collections in their range. These collections are interpreted as argument lists. If a predicate in diagrams is a function, this is often indicated by "`()`" added to the predicate identifier.

For function predicates the black solid arrows in diagrams need a special interpretation. The box at the start of the arrow still denotes the domain of the predicate/function. The instance used here is also the one that receives the result value of the function. The box at the tip of the arrow denotes arguments used in the function's argument list. For example:



This diagram states that any of the four functions from the `log:` namespace can be used as predicate to make a statement. This statement has an instance of `icm:Expectation` as subject. The function delivers its result as value of this instance. This means that the used instance of `icm:Expectation` will gain an implicit `rdf:value` statement with the function result as object.

The diagram also states that one or potentially several instances of `icm:Condition` can be used as function arguments. The statement

```
ex:E1 log:allOf ( ex:C1 ex:C2 ) .
```

is a valid statement according to this diagram.

7. TM Forum Intent Ontology

7.1. The TM Forum Intent Ontology (TIO)

The TM Forum Intent Ontology is a collection of models that allow the expression of intent. It consists of distinct models that address distinct use cases and intent expression concerns. When assembling an intent expression, this means that the vocabulary and semantics used is typically contributed from several distinct models within the TIO. This chapter introduces the already defined parts of the TIO.

Each distinct model is uniquely identified by an IRI. This IRI also serves as namespace identifier for all vocabulary specified as part of the model. Furthermore, model representations in TURTLE format use namespace prefixes to enhance readily of the model expression. The prefix is a substitute for the model and namespace identifier IRI. For example, the intent common model is identified by the IRI "http://tio.models.tmforum.org/tio/v3.6.0/IntentManagementOntology". The class "intent" as specified within this model is then fully identified by appending it to the model and namespace identifier: "http://tio.models.tmforum.org/tio/v3.6.0/IntentManagementOntology/Intent". In practical TURTLE expressions, the model and namespace identifier would be substituted with a prefix definition. The convention and recommendation is to use the prefix "icm" for the intent common model. With the defined prefix the same identifier for the class Intent can be expressed as "icm:Intent". The following chapters state the IRI identifier of the latest published version of a model, the recommended prefix to be used and the specification documents that explain the model vocabulary and semantics.

7.2. TIO Release 3.6

The following table states the scope of the TM Forum Intent Ontology release v3.6.0.

The table also states dependencies between model. One model is depended on another one, if it is using vocabulary from the other model's namespace to express its specification. Only direct, but no indirect dependencies are stated. Indirect dependencies are dependencies that occur if a model is using vocabulary from a second namespace, which in turn is using vocabulary from a third namespace. The dependency is indirect if the dependent model is not directly using vocabulary from the third namespace as well.

In some cases examples are given that use additional vocabulary. Examples do not constitute a dependency unless the vocabulary is an essential part of the normative specification.

All models within the TIO depend on RDF, RDF Schema and XML Schema Definition. Also, Dublin Core and SKOS are used in the formal model expression in TURTLE and JSON-LD. These are not listed explicitly as dependencies.

http://tio.models.tmforum.org/tio/v3.6.0/TIO/

Model	Version	prefix	Specifications	Dependency	Comment
Intent Management Ontology	v3.6.0	imo	[tr292a-imo] [tr292b-imo]		

Model	Version	prefix	Specifications	Dependency	Comment
Function Definition Ontology	v3.6.0	fun	[tr292c-fun]		
Quantity Ontology	v3.6.0	quan	[tr292d-quan]	fun	
Conditions and Logical Operators	v3.6.0	log	[tr292e-log]	fun	
Set Ontology	v3.6.0	set	[tr292f-set]	fun	
Metrics and Observations	v3.6.0	met	[tr292g-met]		
Mathematical Functions Definition and Collection	v3.6.0	mf	[tr292h-mf]	fun	
Security Ontology	v3.6.0	sec	[tr292i-sec]		pre-production
Connector Model	v2.0.0	conn	[tr293-connector]		pre-production
Intent Specification	v3.6.0	insp	[tr299-specification]	imo, icm	
Intent Common Model	v3.6.0	icm	[tr290a-icm] [tr290b-report]	imo, fun	
Intent Validity	v3.6.0	iv	[tr291a-validity]	icm	
Intent Probing	v3.6.0	pro	[tr291b-probe]	icm	
Proposal of Best Intent	v3.6.0	pbi	[tr291c-best]	icm	
Preference of Intent Handling Outcomes	v3.6.0	pre	[tr291g-preference]		
Intent Guarantee	v3.6.0	ig	[tr291h-guarantee]	imo	pre-production
utility	v3.6.0	ut	[tr291i-utility]	mf	

7.2.1. TIO Identification

The TM Forum Intent Ontology as a whole is identified by
IRI: http://tio.models.tmforum.org/tio/v3.6.0/TIO/

7.2.2. Modelling Foundation

The models within the TM Forum intent ontology are build on top of the Resource Description Framework (RDF) specified by the World Wide Web Consortium (W3C). These models are not considered to be part of the TM Forum Intent ontology, but models from the TIO depend on them.

7.2.2.1. Resource Descriptor Framework (RDF)

Identifier (latest version): http://www.w3.org/1999/02/22-rdf-syntax-ns#
recommended namespace prefix: rdf
Specifications: RDF 1.1. Primer [rdf11-primer], RDF 1.1 Concepts and Abstract Syntax [rdf11-concepts], RDF 1.1 Semantics [rdf11-mt]

The Resource Description Framework (RDF) is a framework for representing information in the Web.

7.2.2.2. RDF Schema

Identifier (latest version): http://www.w3.org/2000/01/rdf-schema#
recommended namespace prefix: rdfs
Specifications: RDF 1.1 Schema [rdf11-schema]

RDF Schema provides a data-modelling vocabulary for RDF data. RDF Schema is an extension of the basic RDF vocabulary.

7.2.2.3. XML Schema Definition (XSD)

Identifier (latest version): http://www.w3.org/2001/XMLSchema#
recommended namespace prefix: xsd
Specifications: XML Schema Part 0: Primer Second Edition [xsd-0primer], W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures [xsd11-structures], W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes [xsd11-datatypes]

The XML Schema Definition (XSD) specifies how to formally describe the elements in an Extensible Markup Language (XML) document. In the RDF based intent modelling it primarily provides datatypes.

7.2.3. Other externally defined models

7.2.3.1. Time Ontology in OWL

Identifier (latest version): http://www.w3.org/2006/time#
recommended namespace prefix: t
Specifications: Time Ontology in OWL [owltime]

OWL-Time is an OWL-2 DL ontology of temporal concepts, for describing the temporal properties of resources in the world or described in Web pages. The ontology provides a vocabulary for expressing facts about topological (ordering) relations among instants and intervals, together with information about durations, and about temporal position including date-time information. It is used within the TIO to describe time. It implies a dependency to OWL. In future versions of TIO this dependency is planned to be removed in future versions of TIO by defining a distinct time ontology.

7.2.3.2. Simple Knowledge Organization System (SKOS)

Identifier (latest version): http://www.w3.org/2004/02/skos/core#
recommended namespace prefix: skos
Specifications: SKOS Simple Knowledge Organization System Reference [skos-ref]

The Simple Knowledge Organization System (SKOS), is a common data model for sharing and linking knowledge organization systems via the Web. It is used for adding descriptions to the constituent models within the TIO.

7.2.3.3. Dublin Core Metadata

Identifier (latest version): http://purl.org/dc/terms/
recommended namespace prefix: dct
Specifications: DCMI Metadata Terms [dcmi-terms]

The Dublin Core vocabulary, also known as the Dublin Core Metadata Terms, is a general purpose metadata vocabulary for describing resources of any type. It was first developed for describing web content in the early days of the World Wide Web.

7.2.4. Base Ontology Model:

Models within the intent ontology introduce vocabulary and concepts about intent management as well as general concepts as foundation for expressing intent. The intent ontology defines multiple distinct namespaces dedicated to specific modelling concerns.

7.2.4.1. Intent Management Ontology

Identifier (latest version):
http://tio.models.tmforum.org/tio/v3.6.0/IntentManagementOntology/
recommended namespace prefix: imo
Specifications: TR292A Intent Management Elements [tr292a-imo], TR292B Intent Management State Machines [tr292b-imo]
Status: production

The Intent Management Ontology consists of a specification of intent management elements and intent management state machines.

The intent management elements model is an ontology within the TM Forum intent ontology. It introduces basic terminology used to describe intent based operation. The TM Forum Autonomous Networks Project introduces concepts around the use of intent within an autonomous network. For example, it introduces an architecture and functions to participate in intent based operation, roles within intent life cycle management and the intent API.

The intent management state machines model introduces a generic state machine describing the status of intent handling and the compliance of the system to the requirements expressed by an intent. The intent management state machines model introduces a standard vocabulary to be used between the intent handler and intent owner to communicate the intent handling status and the stage in the intent life cycle. The model defines a generic state machine of intent handling. States reflect compliance to the intent requirements and stages in the intent life cycle. State transitions serve as events that indicate that a significant change in the intent handling status has occurred. A separate, but associate state machine is defined for communicating the progress of intent updates.

7.2.4.2. Function Definition Ontology

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/FunctionOntology/
recommended namespace prefix: fun
Specifications: TR292C Function Definition Ontology [tr292c-fun]
Status: production

The function definition ontology introduces functions as elements in intent expression. Functions are a versatile concept. They can be used, for example, for defining logical or numerical operators or to define the creation and modification of data structure. Furthermore, they improve intuition when working with the intent models. This document provides an ontology and vocabulary for definition and formal description of functions within the TM Forum Intent Ontology (TIO). This is the base for modeling dedicated functions for diverse use cases by other models within the TIO.

7.2.4.3. Quantity Ontology

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/QuantityOntology/
recommended namespace prefix: quan
Specifications: TR292D Quantity Ontology [tr292d-quan]
Status: production

This document defines quantities and a set of operator functions that allow to define, modify, combine and compare quantity objects. Defining goals based on KPIs and metrics is a common task in intent expression. They often consist of a numerical value in combination with a unit. Quantities, as defined in this document are exactly that: a class with properties representing a numerical literal in combination with a unit expression. This document also defines operations for constructing, modifying and combining quantities. Further operators allow comparing and derive inference from quantities.

7.2.4.4. Conditions and Logical Operators

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/LogicalOperators/
recommended namespace prefix: log
Specifications: TR292E Conditions and Logical Operators [tr292e-log]
Status: production

The ontology model specified in this document defines introduces a set of logical operators available for expressing logical relationships in intent modeling.

7.2.4.5. Set Operators

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/SetOperators/
recommended namespace prefix: set
Specifications: TR292F Set Operators [tr292f-set]
Status: production

This document defines an ontology model of set operators. Requirement specification in intents often requires formulating conditions based on sets of things. For example, the state of a network resource shall be one of a set of “good” states in order for the system to meet the requirement. Another example is the requirement that a resource must not be allocated within certain distinct regions or sites. All this involves sets and logic regarding intersection of sets and membership of individuals within a set.

7.2.4.6. Metrics and Observations

Identifier (latest version):
http://tio.models.tmforum.org/tio/v3.6.0/MetricsAndObservations/
recommended namespace prefix: met
Specifications: TR292G Metrics and Observations [tr292g-met]
Status: production

This document defines an ontology model for defining and using metrics and managing observations, such as measurements concerning the metrics.

7.2.4.7. Mathematical Functions Definition and Collection

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/MathFunctions/
recommended namespace prefix: mf
Specifications: TR292H Mathematical Functions Definition and Specification [tr292g-met]
Status: production

This document defines a set of mathematical functions and vocabulary to manage value ranges and composite functions.

7.2.4.8. Security Ontology

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/SecurityOntology/
recommended namespace prefix: sec
Specifications: TR292I Security Ontology [tr292i-sec]
Status: pre-production

The ontology model defined in this document defines the necessary vocabulary that can be used in the security domain. The focus of this document is the specification of how a security intent is expressed.

7.2.4.9. Connector Model

Identifier (latest version): http://tio.models.tmforum.org/tio/v2.0.0/ConnectorModel/
recommended namespace prefix: conn
Specifications: Connector Model v2.0.0 [tr293-connector]
Status: pre-production and not fully aligned with TIO v3.6.0

A connector model introduces a proxy IRI for artifacts from other models that use different identification schemes.

7.2.5. Intent Common Model

The intent common model defines the core vocabulary for expression of intent and intent reports. It specifies the mandatory vocabulary to be supported by intent management functions.

7.2.5.1. Intent Common Model

Identifier (latest version): http://tio.labs.tmforum.org/tio/v1.0.0/IntentCommonModel/
recommended namespace prefix: icm
Specifications: TR290A Intent Common Model - Intent Expression, TR290B Intent Common Model - Intent Reporting
Status: production

The intent common model contains the mandatory vocabulary for the expression of intent and intent reports.

7.2.6. Intent Extension Models for intent Expression

7.2.6.1. Intent Validity

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/IntentValidityOntology/
recommended namespace prefix: iv
Specifications: TR291A Intent Validity - Intent Extension Model [tr291a-validity]
Status: production

With the intent validity model, the intent owner can delegate the validity control to the handler by specifying the detailed conditions of validity within the intent. The model allows defining complex validity schemes involving multiple types of conditions and alternative requirements with mutually exclusive validity.

7.2.6.2. Intent Probing

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/IntentProbing/
recommended namespace prefix: pro
Specifications: TR291B Intent Probing - Intent Extension Model [tr291b-probe]
Status: production

Probing of intent is a dialog between an intent owner and intent handler for the purpose of feasibility and potential outcome assessment of intent requirements. It allows the intent owner to send an intent to an intent handler asking for a prediction of the outcome and fulfillment. In this respect the intent handler is not obliged to actually meet the requirements. It is rather asked to provide an estimate of what it would achieve hypothetically, if it would need to meet the requirements within the intent. This provides the intent owner with valuable information regarding what it can expect if sending this intent for operation asking for actually complying to the intent.

Intent probing is optional and can therefore only be used between an intent owner and an intent handler, it both support it. Intent model vocabulary is provided in this intent extension model rather than being part of the mandatory intent common model.

7.2.6.3. Proposal of Best Intent

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/IntentValidityOntology/
recommended namespace prefix: pbi
Specifications: TR291C Proposal of Best Intent - Intent Extension Model [tr291c-best]
Status: production

This document specifies the intent and intent report vocabulary for proposal of best intent procedure on an intent API. It allows asking an intent management function to state a version of a requirement it can successfully handle. What the intent handler proposes constitutes the most demanding version of the requirement, for which the intent handler would still be able to reach compliance. It means that the intent handler would have solutions and actions available with positive prognosis for compliance to all requirements within the intent including the proposed elevated requirement. Proposal of best intent is an optional intent negotiation procedure between two intent managers within the investigation phase or operation phase of the intent life-cycle. It allows an intent owner to investigate what an intent handler is able to achieve. This can be used before the intent is requested for operation or exploration. Alternatively this procedure can be used while an intent is in operation or exploration to monitor the compliance abilities of the intent handler with respect to certain requirements.

7.2.6.4. Preference of Intent Handling Outcomes

Identifier (latest version):
http://tio.models.tmforum.org/tio/v3.6.0/PreferenceOfHandlingOutcomes/
recommended namespace prefix: pre
Specifications: TR291G Preference of Intent Handling Outcomes - Intent Extension Model [tr291g-preference]
Status: production

This intent extension model specifies the vocabulary that enables an intent handler to request a statement of preference for a potential outcome from the owner of an intent. The intent handler would generate intent reports representing expected outcomes of the solution alternatives under consideration. This way it can communicate the potential outcomes to the intent owner and request a statement of preference. This model also specifies how the intent owner can convey its preferences regarding the presented alternative outcomes back to the intent handler.

7.2.6.5. Intent Guarantee

Identifier (latest version):
http://tio.models.tmforum.org/tio/v3.6.0/IntentGuaranteeOntology/
recommended namespace prefix: ig
Specifications: TR291H Intent Guarantee - Intent Extension Model [tr291h-guarantee]
Status: pre-production

The Intent Guarantee model extends the vocabulary of the TM Forum Intent Ontology (TIO) to support the request for guarantees that are future looking and that are statements of enduring compliance.

7.2.6.6. Utility

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/Utility/
recommended namespace prefix: ut
Specifications: TR291I Utility - Intent Extension Model [tr291i-utility]
Status: production

The utility intent extension model specified in this document introduces optional utility information to be added into an intent. Utility information specifies a function to be used to map the values of a metric into a utility score. This utility score is a measure of business value. Having this information in the intent and associated with requirements expressed with the same metrics allows an intent owner to convey its notion of business value and its preferences to the intent handler. It is doing so in a quantifiable way that is directly enabling calculations within the intent handler for making optimization and conflict resolution decisions.

7.2.7. Intent Management Capabilities and Specification

7.2.7.1. Intent Specification

Identifier (latest version): http://tio.models.tmforum.org/tio/v3.6.0/IntentSpecification/
recommended namespace prefix: insp
Specifications: TR299 Intent Specification [tr299-specification]
Status: production

An intent specification defines rules and constraints regarding allowed content of intent. This allows generating and verifying well-formed intents. Many factors can be considered, such as the nature of an autonomous domain, the services and resources exposed by it and capabilities and properties of intent handlers. Intent specifications can be associated with product and service specifications within catalogs.

8. Intent management features and procedures

This section explains how typical intent expression use cases can be solved using the TM Forum intent Ontology. It explains how to use the model vocabulary. The presented use case in this section will be expanded over time to cover further features of the TIO.

The examples use the latest version of TIO. In this revision of the intent user guide, TIO version 3.6.0 is used.

The RDF statements in the examples are using TURTLE notation. If intent expressions in other notations, such as JSON-LD are needed, please use conversion tools.

8.1. Expressing a metrics based requirement

Many use cases require expressing requirements based on observable metrics. For example, the requirement is that the latency on a slice within the radio network shall not exceed a stated threshold.

Before the intent can be expressed a couple of questions need to be clarified:

What autonomous domain shall handle the intent?
In this example slice latency is a concern of performance management. In this example it is specifically performance management in RAN. This information helps to identify the correct autonomous domain and its intent manager that shall handle the intent. It needs the capability and responsibility scope for managing this concern and resource. The autonomous domain addressed by the intent is RAN management as subdomain on resource management layer.

What resource need to meet the requirements?
In this example the requirement targets a specific instance of a network slice. In the intent expression, this instance will become a member of a target. In this example the slice is identified by the IRI `http://example.org/Slice001` or `ex:Slice001` when using prefix notation. Resource instances and their unique identifiers are typically available from inventory services. If the resource ID is not yet using IRI identification, an IRI for it can be generated according to the recommendation of connector models. For example, 3GPP specifies now to create and IRI from distinguished named (DN) used in 3GPP object management.

How can intent compliance be observed?
This use case deal with a metrics based requirement. The autonomous domain that is supposed to handle this intent would need the capability to measure slice latency in RAN. It would publish in its intent manager capability profile which metrics it can handle and accept intent for. If a metric that measures RAN slice latency is stated as supported, it can be used. Intent managers that support TIO would use IRI to identify metrics. If the metric is defined with different identification schemes, an IRI can be assumed using connectors. In this example the RAN slice latency shall be identified by the IRI `http://example.com/sliceLatency` or `ex:sliceLatency` in short prefix notation

With this information we can successively build the intent expression.

Step 0: Define namespace prefixes

In order to use the compact prefix notation of TURTLE, all used namespace prefixes need to be declared and defined in the intent expression document. In this example we will need at least the intent common model, RDF and RDFS and an example namespace for created instances.

```
@prefix  icm: <http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/>
.
@prefix  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix   ex: <http://example.org/> .
```

Step 1: Instantiate new intent instance

Intent is an instance of class `icm:Intent`. In this example we create a new intent instance identified by the IRI `ex:I1`. This is a globally unique identifier chosen by the intent owner. The intent expression starts with the statement:

```
@prefix  icm: <http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/>
.
@prefix  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix   ex: <http://example.org/> .

ex:I1
  a icm:Intent
.
```

Step 2: Specify a target

We can now add a target definition to the intent. A target states what resources need to comply to a requirement. In this example it is a slice instance. Targets within an intent are defined using containers. They represent sets of all resources that need to comply to the requirements. We can therefore use the `rdfs:member` property to add the slice instance to the target container.

```
@prefix  icm: <http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/>
.
@prefix  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix   ex: <http://example.org/> .

ex:I1
  a icm:Intent
.
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice001
.
```

Step 3: Specify an expectation

Expectations are sets of requirements within an intent. For a metrics based requirement, an property expectation the right choice.

Expectations need to be associated with targets. These are the targets that specifically need to comply to the requirements of this expectation.

Expectations bear a boolean truth value that states compliance to the requirements. The expectation needs to contribute its compliance state to the overall compliance state of the intent. This is achieved using a logical operator.

We can add these components to the intent:

```
@prefix  icm: <http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/>
.
@prefix  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix   ex: <http://example.org/> .
@prefix  log: <http://tio.models.tmforum.org/tio/v3.6.0/LogicalOperators> .

ex:I1
  a icm:Intent ;
  log:allOf ( ex:E1 )
.
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice001
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1
.
```

This example uses the `log:allOf` function. It constitutes a logical conjunction operator (AND). It results in "`true`", if all its arguments are "`true`". In this example it only has one argument, the expectation `ex:E1`. As truth values of expectations and intent are interpreted as compliance to requirements, this example expresses, that the compliance result for the overall intent depends on the compliance to the expectation `ex:E1`. Also, the namespace for logical operators was added to the prefix definition.

The target ex:T1 was associated with the expectation ex:E1 using the property icm:target. This asserts that the expectation ex:E1 assumes compliance, if the slice instance as member of the target is compliant to the requirements.

Step 4: Specify compliance conditions

So far we have created the structure of the intent with respect to the composition of overall compliance contributions and the resources that shall comply, but we have not yet specified what the actual requirement is. For this purpose we add a condition based on the chosen metric for slice latency. In this example the measured slice latency must be smaller than 100 ms for the slice to be considered to compliant. Further namespace prefixes are added as additional vocabulary is used.

```
@prefix  icm: <http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/>
.
@prefix  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix   ex: <http://example.org/> .
@prefix  log: <http://tio.models.tmforum.org/tio/v3.6.0/LogicalOperators> .
@prefix quan: <http://tio.models.tmforum.org/tio/v3.6.0/QuantityOntology> .
@prefix  met:
<http://tio.models.tmforum.org/tio/v3.6.0/MetricsAndObservations> .

ex:I1
  a icm:Intent ;
  log:allOf ( ex:E1 )
.
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice001
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1
  log:allOf ( ex:C1 )
.
ex:C1
  a log:Condition ;
  quan:smaller ( met:lastValue ( ex:sliceLatency )
                "100 ms"^^quan:quantity )
.
```

The condition `ex:C1` is added. It gets its truth value from the function `quan:smaller`. The `quan:smaller` function constitutes a comparison of two quantities. Here the first quantity is the last value that was observed for the metric `ex:sliceLatency`. The second quantity in the comparison is a literal with the datatype `quan:quantity`. It has the value "`100ms`". Quantities constitute values that can have a unit. Here the numerical value "`100`" is combined with the unit identifier "ms". Overall the quan:smaller function assumes the boolean value "`true`" if, when and while the last measured value for the slice latency is smaller than "`100ms`". This result of the smaller comparison determines the truth value of the condition.

The condition ex:C1 is associated with the expectation ex:E1 through the logical operator function log:allOf. This way they condition contributes its truth value to the value of the expectation. This association also implies that the condition inherits the target as scope. This determines that the slice latency value used here shall be measured for the slice that is a member of the target.

This completes the example use case in scope of this chapter. The resulting intent is however missing subscriptions to intent reports. Adding them is a topic for another use case.

8.2. Requiring intent reports

Intent reports are the means by which an intent owner can stay informed about the compliance to intent requirements achieved by the intent handler. While intent reports are technically optional like any other intent requirement, in practice at least a minimum set of report subscriptions would always be sensible to add to support the intent owner in monitoring its intent. Typically, reports would be required for significant events in the intent life cycle, such as intent acceptance or rejection and the removal of intent. Also reports for losing or gaining compliance are typically requested. Beyond that, the frequency of reporting and the type and the information provided within a report can be steered by the intent owner depending on its needs.

Intent reporting is requested by the intent owner with reporting expectations in the intent expression.

For requiring the needed intent reports, an intent owner needs to address a few concerns and questions?

What content shall be provided in an intent report?
Intent reports can only provide a basic status update by stating the overall compliance, or thy can have detailed information per expectation and individual condition. It can also include observed values for the metrics used in the specification of requirements. An intent owner needs to clarify what level of information it needs at a particular intent handling event. What information in needed from an intent report is steered by the subclass of reporting expectation used.

What intent elements need to be reported about?
Intent reports can be scoped to report about certain intent elements. This steers if information shall be provided within an intent report only for the intent as a whole or for certain instances of intent elements such as certain expectations, conditions or contexts. This is done by specifying the target of a reporting expectation. Information is included for all members of the target associated with a reporting expectation.

This means, the information provided by an intent report can be requested through choosing respective subclasses of reporting expectations in combination with intent elements being specified as members of the reporting expectation's target. In the example presented in this chapter only basic information about the compliance status of the entire intent is requested. This is therefore a minimal intent report request presented as a baseline.

When shall an intent report be created?
Intent reports are created based on intent handling events. An intent reporting expectation would therefore enumerate intent handling events. If any of the enumerated events is raised, the intent handler shall create a new intent report. The intent management ontology specifies a set of predefined intent handling events based on state transitions in the intent management state machine. These events cover many significant points in the intent life-cycle and intent handling progress. They mark, for example, the acceptance or rejection of the intent and the transition from compliance to degradation and back.

Who shall be notified about a new report?
The reporting expectation can explicitly state which functions shall be notified about the intent report. This is called a report destination. Typically, the intent owner itself would be added as report destination, but also additional destinations, such as logging or other monitoring systems can be added.

The following intent expression contains a request for intent reports:

```
@prefix  icm: <http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/>
.
@prefix  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix   ex: <http://example.org/> .
@prefix  log: <http://tio.models.tmforum.org/tio/v3.6.0/LogicalOperators> .
@prefix quan: <http://tio.models.tmforum.org/tio/v3.6.0/QuantityOntology> .
@prefix  met:
<http://tio.models.tmforum.org/tio/v3.6.0/MetricsAndObservations> .

ex:I1
  a icm:Intent ;
  log:allOf ( ex:E1 ex:E2 )
.
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice001
.
ex:T2
  a icm:Target ;
  rdfs:member ex:I1
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1
  log:allOf ( ex:C1 )
.
ex:C1
  a log:Condition ;
  quan:smaller ( met:lastValue ( ex:sliceLatency )
                "100 ms"^^quan:quantity )
.
ex:E2
  a icm:ReportingExpectation ;
  icm:target ex:T2 ;
  icm:reportDestination ex:IMF1 ;
  icm:reportTriggers [ rdfs:member imo:IntentRejected ,
                                  imo:IntentAccepted ,
                                  imo:Degrades ,
                                  imo:Complies
                    ]
.
```

This example intent contains a reporting expectation to require the generation of intent reports for this intent. The target of this reporting expectation is ex:T2 with one member, which is the intent itself. This specifies that the report shall only contain information about the intent as a whole. This means, no detailed reports about elements of the intent are requested.

The intent report has only one destination, which is the intent owner. In this example the intent owner is identified by ex:IMF1. Based on this information the intent handler will issue a notification containing the intent report towards the intent owner. Note that this mechanism might be revised or removed in the coming revisions to be more aligned with notifications on open APIs.

The reporting expectation also specifies report triggers. It is using a container of intent handling events. Whenever one of these events is raised in the intent handler a new intent report shall be created. In this example an intent report shall be created when the intent is accepted or rejected, or if the intent changes states between compliance and degraded.

The following intent report is an example of a report that could be issued according to the example intent and its reporting expectation.

```
ex:IR1
  a icm:IntentReport ;
  icm:reportGenerated [ t:inXSDDateTimeStamp "2023-02-
06T13:30:10+01:00"^^xsd:dateTime ] ;
  icm:reportNumber "4"^^xsd:positiveInteger ;
  icm:about ex:I1 ;
  imo:handlingState imo:StateCompliant ;
  imo:updateState icm:StateNoUpdate ;
  imo:event ex:Complies001 ;
  icm:result true
.
```

The example intent report provides the following information:

- A time stamp of intent report generation. This intent report was created on the 6th of February 2023 at 13:30 and 10 seconds with time zone UTC+1.
- A sequence number of the individual report. This is the 4th report generated for this intent.
- A reference to the intent element the intent report provides information about. Here is the intent.
- The intent handling and intent update state at the time of report generation. The intent handing is in state "Compliant" and there is no ongoing intent update.
- The individual event that triggered the report.
- Compliance information about the intent and each intent element in scope when applicable. This is provided with the property `icm:result`. It shows "true" because the compliance evaluation based on expectations did result in the boolean value "true", which is interpreted as being compliant. This information appears to be redundant as the handling state also indicates compliance. However, the property icm:result can also be used for detailed results of individual intent elements such as expectations and conditions. Using it also for the intent means that it is used consistently for stating the value associated with an intent element.

It is also possible to include reasons that explain the result. This is most significant for reports regarding rejection of intent or results that show degradation. In this example reasons are left out as compliance is reached and reported; thus nothing needs to be explained.

This intent report does not contain further report elements for other intent elements as they were not requested.

8.3. Expressing conditional validity of requirements

An intent requirement might only be required when certain side conditions apply. Furthermore, multiple different variations of a requirement shall be applicable specific to the situation. For example, a slice latency of 100 ms shall only be the required for up to 1000 users being served by the slice. If the number of users increases above 1000, the required latency shall be only 150 ms. Use cases like this can be addressed using validity contexts.

A validity context defines a condition and associates this condition with elements of the intent. It can be applied to an expectation, another condition, or the entire intent. If and while the condition of the validity context evaluates to "false", compliance is no longer expected from the intent element the validity context is associated with.

The following questions need to be answered and clarified for using validity context.

What requirements shall be contextualized?
This clarifies to which elements of an intent the validity context is attached. If multiple requirements are provided by an intent and validity context is only needed for some of them, the validity context would be applied to respective expectations or conditions.

What is the wanted interpretation of valid and invalid requirements?
A validity context conditionally disables requirements. This enforces that the compliance state for these requirements indicates compliance regardless if this actually backed by observation. The point is that invalid requirements must not matter in achieving compliance. In scenarios with complex logical relationships between distinct requirements of the intent caution is needed when validity context is applied. The forced compliance of some requirements can have unintended side effect. For example if two expectations are in a disjunction regarding compliance, this means that the compliance of only one of these requirements would lead to overall compliance. If one of the contributing expectations becomes invalid, its compliance is enforced. Due to the disjunction this enforces compliance for the entire intent. This might not be the wanted result of invalidating one expectation. If the wanted result is that only the remaining valid expectation determines compliance while the other expectation is invalid, a different expression is needed.

How is the validity evaluated?
This means what metrics and observations are needed to evaluate validity. Like any other metrics based condition in the intent, the intent handler needs to be able to observe and obtain values for the used metrics. If the needed metrics are supported, they can be used in the intent.

This example use case in this chapter demonstrates an intent that uses validity context for alternative latency requirements. Depending on the number of users served by a slice, a different latency requirement shall apply.

```
@prefix  icm: <http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/>
.
@prefix  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix   ex: <http://example.org/> .
@prefix  log: <http://tio.models.tmforum.org/tio/v3.6.0/LogicalOperators> .
@prefix quan: <http://tio.models.tmforum.org/tio/v3.6.0/QuantityOntology> .
@prefix  met:
<http://tio.models.tmforum.org/tio/v3.6.0/MetricsAndObservations> .
@prefix   iv: <http://tio.models.tmforum.org/tio/v3.6.0/IntentValidityOntology> .

ex:I1
  a icm:Intent ;
  log:allOf ( ex:E1 )
.
ex:T1
  a icm:Target ;
  rdfs:member ex:Slice001
.
ex:E1
  a icm:PropertyExpectation ;
  icm:target ex:T1
  log:allOf ( ex:C1 ex:C2 )
.
ex:C1
  a log:Condition ;
  quan:smaller ( met:lastValue ( ex:sliceLatency )
                "100 ms"^^quan:quantity ) ;
  iv:validIf ex:VC1
.
ex:C2
  a log:Condition ;
  quan:smaller ( met:lastValue ( ex:sliceLatency )
                "150 ms"^^quan:quantity )
.
ex:VC1
  a iv:Validity ;
  a log:Condition ;
  quan:smaller ( met:lastValue ( ex:NumberOfUsersOnSlice )
                [ rdf:value 1000 ]
              )
.
```

This example shows an intent with two conditions contributing the compliance of a property expectation. Both conditions use the same metric, but introduce different thresholds. This means, if the lower threshold condition is true, the one with the higher threshold is also true. As both are in a conjunction, this means that the overall compliance of the expectation is effectively only determined by the lower threshold.

This intent introduces the validity context `ex:VC1`. It indicates validity of the observed number of users on the slice is smaller than 1000. This validity is assigned to the condition `ex:C1`. This means, `ex:C1` is invalid and forced to value "`true`" if and when the slice serves 1000 or more users. In this case the condition `ex:C1` is forced to true and the condition `ex:C2` with the higher latency threshold becomes the deciding factor for the resulting compliance of the expectation.

9. Terms & Abbreviations Used within this Document

This chapter defines terms used in Autonomous Networks following and extending TM Forum's Autonomous Networks Glossary [ig1258]. Within the intent user guide the focus lies on terminology used in intent based operation.

9.1. Terminology

9.1.1. Autonomous Domain

An autonomous domain identifies a part of an autonomous network. An autonomous domain is typically defined with respect of management scope including management functions and managed resources or services. Autonomous domains typically implement an intent management function to participate in intent based operation.

9.1.2. Autonomous Network

An autonomous network is a network that implements automated and adaptive operation.

9.1.3. Business intent

Business intent specifies requirements for the business operation layer. This means that intent managers allocated in business operation would receive and handle this intent.

9.1.4. Context

Context is an object within the intent expression. It defines scoping, constraints or interpretation hints for requirements within the intent.

9.1.5. Expectation

An expectation is an object within the intent expression. An Expectation defines a subset of requirements. Targets are assigned to expectations and therefore expectations associate sets of requirements to entities that need to meet these requirements.

9.1.6. Intent

Intent is an object that contains requirements for an autonomous domain. It is an instance of class icm:Intent.

9.1.7. Intent API

An API that allows life-cycle management of intent. It enables sending intent information, modification or removal of intent and the communication of intent reports. Intent managers in the roles of intent handler and intent owner communicate with each other through an intent API.

9.1.8. Intent Applicability

An intent is applicable if its expression is formally correct and in line with the capabilities of its intent handler. This means the intent expression does not contain any errors that would make the intent not understandable by the intent handler. If an intent handler is receiving an intent it cannot understand, it would reject the intent. An intent handler would consider an intent not applicable, if the intent is either using invalid expressions or expressions that require support for optional models that this intent handler does not implement. This means the intent handler's capabilities were not respected when assembling the intent expression. If an intent expression is correct and within the limits of the handler's capability, it is applicable regardless if the requirements within the intent can be achieved. Success of intent compliance is a concern of intent feasibility and intent guarantee, but not part of intent applicability.

9.1.9. Intent Assurance

Intent assurance is concerned with the functions that are necessary to ensure that the network indeed complies with the desired intent. Intent Assurance is also a phase in intent handling. It starts after intent fulfillment. Typically, intent assurance consists of continuous observation and monitoring by the intent handler if the autonomous domain complies with intent expectations. If the intent handler detects degradation or an opportunity to improve the system state, it would take healing or optimization actions.

9.1.10. Intent-Based Network (IBN)

A network that can be managed and operated using intent.

9.1.11. Intent-Based Networking

Managing and operating a network by following the paradigm of setting requirements using intent.

9.1.12. Intent-Based System

A system that supports management functions that can be guided using intent.

9.1.13. Intent-Based Operation

Operating a system by following the paradigm of setting requirements for the network using intent.

9.1.14. Intent Common Model

An intent common model specifies domain-independent generic modeling artifacts such as the intent class and expectation class. Intent objects contain a set of expectations, which are distinct and diverse types of requirements allowing to address all relevant concerns

9.1.15. Intent Compliance

Compliance to an intent means that the operated network or system meets the requirements stated within an intent. It is the main objective on intent managers to reach compliance to all intents within their autonomous domain. They typically achieve this by utilizing the resources provided by the network. They can, for example, order suitable services, configure network functions or delegate to further intent managers by sending further intent. Being compliant is also a state in the intent handling state machine defined as part of the TIO. This state is assumed for an intent, if the all its requirements as specified by expectations are met. The autonomous domain is considered degraded with respect to an intent if it is not or only partially meeting the specified requirements. The compliance status is also a key information element shared within intent reports. In this respect compliance can also be stated individually per expectation.

9.1.16. Intent Expression

The expression of all elements of an intent. This includes, for example, expectations, context, targets and other information elements within an intent. An Intent contains a set of expectations expressed in statements using a particular expression language such as Turtle or JSON-LD.

9.1.17. Intent Extension Model

Intent extension models are vocabularies that are designed to be used in conjunction with the Intent Common Model. They "extend" the Intent Common Model and introduce additional expressiveness and modeling features for creating intent expressions. There are several ways that intent extension models can be used. For example, they can be used to add domain-specific knowledge or expertise to a model, to enable the model to handle a wider range of input data or use case scenarios, or to improve the model's performance on a particular task or problem.

9.1.18. Intent Feasibility

An intent is feasible if the autonomous domain of the intent handler is able to comply to its requirements and successfully fulfill and assure them. If an intent is feasible is a key question in intent negotiation. This means an intent handler might be required to state possible feasibility before it is requested to actually try to reach compliance. For an intent to be feasible it is required that it is applicable as well. Intent feasibility is assessed in addition to applicability by including operational concerns, such as the availability of sufficient resources. If an intent handler states that an intent is feasible, this reflects the situation at the point in time when it is assessed. It is not a commitment by the intent that the intent would still be feasible in the future. Such a commitment would be addressed by intent guarantees. A synonym of intent feasibility would be intent viability.

9.1.19. Intent Fulfillment

Intent fulfillment is concerned with the functions that take intent from its origin to its realization in the network. In intent handling there is a fulfillment phase concerned with the initial actions taken to reach compliance. Fulfillment ends after these initial actions were executed leading to the intent handling phase of intent assurance. Fulfillment was successful if it reaches compliance.

9.1.20. Intent Guarantee

A guarantee in the context of intent handling refers to a confirmation by an intent handler that it will be able to comply to intent requirements. An intent guarantee is characterized by the time until the guarantee expires and the probability that the guarantee holds within this timeframe. This means a guarantee has a "best before" date and there is still a chance that the system degrades regardless of the guarantee. However, a guarantee allows controlling the probability of failure. An intent owner can request a guarantee from an intent handler and the intent handler would confirm the requested level of guarantee, or if it is not able to provide this guarantee, it can state what level of guarantee it would be able to provide. Furthermore, the intent interface allows regular and automated extensions of the guarantee period.

9.1.21. Intent Handler

Intent handler is a role of an intent manager in the life-cycle management of an intent. An intent owner is the receiver of intent and the creator of intent reports for the intents it has received. Therefore, an intent handler considers the requirements, goals and constraints specified in the received intent, when operating the autonomous domain and system it is responsible for. An intent handler typically implements monitoring of compliance to intent expectations and action taking to reach and preserve compliance to the intent expectations. For every distinct intent there is exactly one intent handler and one intent owner. An Intent manager in the role of intent handler does not modify or remove intent. Only the intent owner can initiate these operations. The intent handler is responsible for creating intent reports to notify the intent manager in the role of intent owner and potentially other authorized functions about the status of intent handling including information about compliance.

9.1.22. Intent Interface

An intent interface is a communication interface or interface layer that is designed to facilitate the exchange of intents between autonomous domains or autonomous networks. It enables systems to communicate their goals, constraints, and requirements to one another, and to negotiate and resolve conflicts or discrepancies between these intents.

9.1.23. Intent Management Function

The Intent Management Function (IMF) is the management function of an autonomous domain that is responsible for handling intent requests and for translating that intent into specific actions for the domain to perform. An intent management function can assume the role of intent handler or intent owner. For every intent there is exactly one intent owner and exactly one intent handler. Therefore, intent is used in the communication of expectations between two instances of intent management functions. One in the role of intent owner and the other in the role of intent handler. However, a given instance of an intent management function can be in the role of intent owner for multiple intents and at the same time in the role of intent handler for a different set of intents.

9.1.24. Intent Manager

An intent manager refers to an instance of an intent management function.

9.1.25. Intent Negotiation

Intent negotiation refers to interactions over the intent interface between two instances of intent management functions. In this process they can assess feasibility of intent, observe possible intent handling results, propose different expectations or provide preferences and feedback regarding potential solution outcomes. Therefore, the term intent negotiation refers to a set of distinct features and capabilities on the intent interface. Intent negotiation enables improved decision-making for resolving conflicts or finding optimal solutions. It's an iterative process of request-reply interactions between intent management functions. Intent negotiation is an important aspect of autonomous systems, as it allows systems to adapt and respond to changing circumstances or unexpected challenges, and to coordinate their actions with other systems or agents in order to achieve their goals.

9.1.26. Intent for Exploration

Intent can be requested for exploration. This means the intent owner sends and request to the intent handler over the intent interface, but it is not asking to actually meet the requirements. Instead, the intent handler would assess the intent and simulate its operation. An intent sent for exploration will therefore not lead to actions that effect the operation of the autonomous domain of the intent handler. No resources are supposed to be sized or configurations changed. Instead, the intent handler would execute its intent handling in a sandbox environment simulating the operation. This is a key component of intent negotiation, because it allows an intent owner to observe the potential behavior of the intent handling results its requirements would achieve, including potential collateral effects. Therefore, intent for operation and intent for exploration are fundamental modes of operation of an intent handler with respect to an intent. The intent owner can request transition between exploration and operation modes. Supporting intent for exploration is an optional capability of intent handlers.

9.1.27. Intent for Operation

Intent can be requested for operation. This means an intent owner sends a request to the intent handler over the intent interface asking to meet the intent expectations. This mean the intent handler is now obliged to find and implement solutions that would achieve compliance to the intent expectation. This is in contrast to intent for exploration. Therefore, intent for operation and intent for exploration are fundamental modes of operation of an intent handler with respect to an intent. The intent owner can request transition between exploration and operation modes.

9.1.28. Intent Orchestration

A set of functions that deals with the actual configuration and provisioning steps that need to be orchestrated across the network for reaching compliance to an intent.

9.1.29. Intent Owner

Intent owner is a role of an intent manager in the life-cycle management of an intent. An intent manager in the intent owner role is the originator of intent. It is the intent manager of an autonomous domain that needs another autonomous domain to meet certain expectations. It is using intent to express these expectations to be communicated via an intent interface to the intent management function of the targeted autonomous domain. The intent owner is responsible for managing the intent's life cycle. Consequently, the intent owner is the only entity and role allowed to modify intent. It is also responsible to actively remove the intent if the requirements it contains are no longer required.

9.1.30. Intent Management Function

An intent management function is a software entity involved in intent based operation. Every intent enabled autonomous domain implements an instance of an intent management function. It is the single point of contact for all matters regarding intent within an autonomous domain. It implements an intent API.

9.1.31. Intent Manager

An intent manager is a software entity involved in intent based operation. It is an instance of an intent management function. It has the role of intent owner or intent handler for an intent. It implements an intent API. It is the single point of contact or all matters regarding intent within an autonomous domain.

9.1.32. Intent Owner

An intent owner is a role of an intent manager in the life-cycle management of an intent. An intent owner is the source of an intent. It is a function that determines requirements an autonomous domain needs another autonomous domain to comply to and uses intent to communicate, manage and monitor the compliance to these requirements. An intent manager is also authorized to modify and remove intent.

9.1.33. Intent Report

An intent report is an object that allows intent management functions to communicate the state and success in handling an intent. Typically, an intent report is generated by an in intent manager in the role of intent handler and received by an intent manager in the role of intent owner. However, also other and additional receivers of intent reports are possible. Intent reports are also associated with an intent they report about.

9.1.34. Intent Request

An intent request is the communication of an intent from an intent owner to an Intent handler over an intent interface. Intent requests can also involve modification or removal of intent. In the Tm Forum intent ontology an intent request distinguished intent sent for operation vs. intent sent for exploration.

9.1.35. Intent Response

An intent response is the response of an intent handler to intent requests in the form of intent reports.

9.1.36. Knowledge Graph

A knowledge graph is a model of real-world entities and their relationships, typically represented as a graph, that is used to represent and organize large amounts of structured and unstructured data in a way that is both human-readable and machine-readable. It is a comprehensive representation of a domain or a set of related domains that contains both data and the relationships between the data elements. It is designed to facilitate the representation, management, and processing of knowledge in a machine-readable format.

9.1.37. Model Federation

Model federation refers to the process of combining or integrating multiple models into a single, larger model. This can be done for a variety of reasons, such as to enable a model or ontology to handle a wider range of expressions (greater expressiveness) from multiple domain-specific models. This concept is used in the TM Forum's Intent Ontology (TIO). The intent modeling in TIO recommends expressing intents from a federation of models. While the intent common model contains general and domain-independent aspects any number of intent expansion 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 needs to understand and what the intent objects addressing this domain can express.

9.1.38. Resource Intent

Resource intent specifies requirements for the resource operation layer. This means that intent managers allocated in resource operation would receive and handle this intent.

9.1.39. Service Intent

Service intent specifies the requirements for the service operation layer. This means that intent managers allocated in service operation would receive and handle this intent.

9.1.40. Target

Targets are sets of things (e.g. resources, services, functions, nodes ...) that need to comply to requirements expressed within an intent. Targets are associated with expectations within an intent.

9.1.41. Utility

In the context of autonomous systems, utility refers to the value or usefulness of a particular system or component to a user, or to the overall network. It is often used to measure the effectiveness or efficiency of a system or component in achieving a particular goal or meeting a specific need. As such, the AN project refers to utility as an aspect of intent as it describes "knowledge about what makes an outcome or situation preferential" (IG1253). In the context of intent modeling and expression, utility refers an optional information object associated with expectations and their metrics. Utility information coveys information how preferential and valuable a reached result for the expectation according to its metrics are. This allows the intent handler to assess the relative value of achieved outcomes or the value of solutions that would reach these outcomes. This can be a key tool for optimization and conflict resolution decisions.

9.2. Abbreviations

Abbreviation/Acronym	Abbreviation/Acronym Spelled Out
3GPP	3rd Generation Partnership Project
AD	Autonomous Domain
AI	Artificial Intelligence
AN	Autonomous Network
ANP	Autonomous Networks Project
API	Application Programming Interface
b-node	Blank Node
CSP	Communication Service Provider
DCMI	Dublin Core Metadata Initiative [dcmi]
DN	Distinguished Name
DSL	Domain Specific Language
sTOM	enhanced Telecom Operations Map (aka. Business Process Framework)
IBN	Intent Based Network
IEC	International Electrotechnical Commission [iec-home]
IETF	Internet Engineering Task Force [ietf]
IG	Introductory Guide
IMF	Intent Management Function
IRI	Internationalized Resource Identifier [iri]
IRTF	Internet Research Task Force [irtf]
ISO	International Organization for Standardization [iso-home]
ISQ	International System of Quantities
JSON	JavaScript Object Notation [json]
JSON-LD	JSON-Linked Data [jsonld]

Abbreviation/Acronym	Abbreviation/Acronym Spelled Out
MEF	Metro Ethernet Forum (aka. MEF Forum) [mef-home]
ML	Machine Learning
TURTLE	Terse RDF Triple Language [turtle]
UML	Unified Modeling Language [uml2]
URI	Uniform Resource Identifier [uri]
URL	Uniform Resource Locator
OWL	Web Ontology Language [owl2doc]
RAN	Radio Access Network
RDF	Resource Description Framework [rdf11-primer]
RDFS	Resource Description Framework Schema [rdf11-schema]
SDO	Standards Defining Organization
SID	Shared Information and Data model (aka. Information Framework) [sid-home]
SKOS	Simple Knowledge Organization System
SLA	Service Level Agreement
SLO	Service Level Objective
TIO	TM Forum Intent Ontology
W3C	World Wide Web Consortium [w3c]
XML	Extensible Markup Language [xml10]
XSD	XML Schema Definition [xsd-0primer]
YAML	YAML Ain't Markup Language

10.References

[3gpp]	**3GPP - The Mobile Broadband Standard**. URL: https://www.3gpp.org/ Home page of the 3rd Generation Partnership Project (3GPP)
[ai-modern]	S. J. Russell; P. Norvig. **Artificial Intelligence: A Modern Approach (4th Edition)**. Pearson 2020. URL: http://aima.cs.berkeley.edu/ Comprehensive introduction to the theory and practice of artificial intelligence.
[bcp47-language]	A. Phillips; M. Davis. **Tags for Identifying Languages**. IETF Best Current Practice. September 2009. URL: http://tools.ietf.org/html/bcp47 Describes the structure, content, construction, and semantics of language tags for use in cases where it is desirable to indicate the language used in an information object.
[dcmi]	**The Dublin Core Metadata Initiative - Homepage** URL: https://www.dublincore.org/ The Dublin Core Metadata Initiative, or "DCMI", is an organization supporting innovation in metadata design and best practices across the metadata ecology.
[dcmi-terms]	DCMI Usage Board. **DCMI Metadata Terms**. DCMI Recommendation. 2020-01-20. URL: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/ This document is an up-to-date, authoritative specification of all metadata terms maintained by the Dublin Core Metadata Initiative. Included are the fifteen terms of the Dublin Core Metadata Element Set (also known as "the Dublin Core") plus several dozen properties, classes, datatypes, and vocabulary encoding schemes.
[gb922-sid]	**Standards Addenda for Information Framework Suite** v23.5. TM Forum Guidebook GB922. Version 23.5. URL: https://www.tmforum.org/resources/suite/gb922-standards-addenda-for-information-framework-suite-v23-5/ This is a collection of standards regarding the Information Framework (SID). See also the Information Framework Homepage [sid-home].
[iec-home]	**International Electrotechnical Commission**URL: https://www.iec.ch/homepage Homepage of the International Electrotechnical Commission (IEC)
[ig1218-ba]	D. Sun. **Autonomous Networks Business Requirements and Framework**. TM Forum Introductory Guide IG1218. Version 2.1.0. 22 January 2022. URL: https://www.tmforum.org/resources/standard/ig1218-autonomous-networks-business-requirements-and-framework-v2-1-0/

	This document provides business requirements and business framework of services and infrastructure supported by Autonomous Networks, including the user requirements per user stories, key business capabilities and framework, and related key metrics for measuring autonomous levels, as well as new business models of production, ecosystem, collaboration. In addition, examples of the lifecycle of services is illustrated for understanding the usage of business requirements and framework. This document serves as the general guideline for pertinent work streams and work items, including user stories and use cases, technical architecture and interface/APIs specs, PoCs/catalyst projects, testing and verification, as well as industry collaboration.
[ig1230-ta]	K. McDonnell. **Autonomous Networks Technical Architecture**. Tm Forum Introductory Guide IG1230. Version 1.1.1 URL: https://www.tmforum.org/resources/introductory-guide/ig1230-autonomous-networks-technical-architecture-v1-1-1/ This document introduces the technical architecture of autonomous networks. It provides the foundation for the technical work in the Autonomous Networks project at TM Forum.
[ig1251-ra]	K. McDonnell. **Autonomous Networks - Reference Architecture**. TM Forum Introductory Guide IG1251. Version 1.6.1. URL: https://www.tmforum.org/resources/introductory-guide/ig1251-autonomous-networks-reference-architecture-v1-0-1/ This guide introduces the autonomous networks reference architecture and provides realization scenarios instantiating the architecture as described.
[ig1253-intent]	J. Niemöller. **Intent in Autonomous Networks**. TM Forum Introductory Guide IG1253. Version 1.3.0. 15 August 2022. URL: https://www.tmforum.org/resources/how-to-guide/ig1253-intent-in-autonomous-networks-v1-3-0/ The purpose of IG1253 is to document and define intent-driven operation according to the work in the Autonomous Networks project. This includes a definition of intent as well as the role of intent in of autonomous operation and the operational principles it implies. Furthermore, this set of documents will define the interface and API for communicating intent, the life-cycle management of intent objects, and the modeling principles of intent. The introductory guide IG1253 has sub-documents IG1253A-E. They are already replaced by model specifications and open APIs and should not be used anymore. The main document IG1253 will be phased out once the intent user guide [ig1358-ug] is completed. The intent user guide will serve the same role as general introduction to intent in TM Forum, but with updated content and many practical details and recommendations.
[ig1258-glossary]	K. McDonnell. **Autonomous Networks Glossary**. TM Forum Introductory Guide IF1258. Version 1.1.0. 14 April 2023. URL: https://www.tmforum.org/resources/technical-report/ig1258-autonomous-networks-glossary-v1-1-0/ This document provides a glossary of terminology relevant to the published deliverables of the Autonomous Networks (AN) project

	with the goal to achieve a common glossary across all the AN deliverables and to serve as terminology reference for use across the industry. Where necessary, descriptions providing background for formal concise definitions will also be provided. In addition to terms that are introduced in the AN documents, related terms taken or derived from external industry publications, or existing TM Forum deliverables are also included. For a complete listing of AN deliverables please refer to IG1260 AN Project Deliverables Guide.
[ig1259-approaches]	M. Nair. **Study of Telecom Industry Intent Meta-Modeling Approaches**. TM Forum Introductory Guide IG1259, Version 1.0.0. 26 July 2021 URL: https://www.tmforum.org/resources/how-to-guide/ig1259-study-of-telecom-industry-intent-meta-modeling-approaches-v1-0-0/ This study explores the approaches followed by different standardization organizations for intent meta modeling and highlights the readiness, relevance of use as a reference methodology, and the context of usage that identifies what aspect of the SDO artifact can be leveraged. It should be noted that this study does not compare the SDO approaches directly, because the objective, context, and the maturity of intent-driven management specification development in the various SDOs considered are different.
[ig1358-ug]	J. Niemöller. **Intent Based Operation User Guide**. TM Forum Introductory Guide IG1258. Version 1.0.0. URL: ... This is the intent user guide and intent home page based on the same content.
[iri]	M. Duerst; M. Suignard. **Internationalized Resource Identifiers (IRIs)**. IETF Proposed Standard RFC8259. December 2017.URL: https://datatracker.ietf.org/doc/html/rfc3987 This document defines the Internationalized Resource Identifier (IRI), as a complement to the Uniform Resource Identifier (URI) [uri].
[ietf]	**Internet Engineering Task Force** URL: https://www.ietf.org/ Home page of the Internet Engineering Task Force (IETF)
[irtf]	**Internet Research Task Force** URL: https://www.irtf.org/ Home page of the Internet Research Task Force (IRTF). The Internet Research Task Force (IRTF) focuses on longer term research issues related to the Internet while the parallel organization, the Internet Engineering Task Force (IETF), focuses on the shorter term issues of engineering and standards making.
[iso-home]	**ISO: Global standards for trusted goods and services** URL: https://www.iso.org/home.html Homepage of the International Organization for Standardization (ISO)
[iso8000]	Standards by ISO/TC12 **Quantities and Units** URL: https://www.iso.org/committee/46202/x/catalogue/

	ISO/IEC 80000 is multipart standard formalizing the international system of quantities and the International System of Units.
[json]	T. Bray, Ed. **The JavaScript Object Notation (JSON) Data Interchange Format**. IETF Internet Standard RFC3986. January 2005. URL: https://datatracker.ietf.org/doc/html/rfc8259 JavaScript Object Notation (JSON) is a lightweight, text-based, language-independent data interchange format. JSON defines a small set of formatting rules for the portable representation of structured data.
[json-ld]	G. Kellogg; P.-A. Champin; D. Longley. **JSON-LD 1.1, A JSON-based Serialization for Linked Data.** W3C Recommendation. 16 July 2020. URL: https://www.w3.org/TR/json-ld11/ This specification defines JSON-LD 1.1, a JSON-based format to serialize Linked Data. The syntax is designed to easily integrate into deployed systems that already use JSON, and provides a smooth upgrade path from JSON to JSON-LD. It is primarily intended to be a way to use Linked Data in Web-based programming environments, to build interoperable Web services, and to store Linked Data in JSON-based storage engines.
[mef-home]	**MEF Forum** URL: https://www.mef.net/ Homepage of the MEF Forum
[oda-home]	**Open Digital Architecture**URL: https://www.tmforum.org/oda/ Homepage of the Open Digital Architecture (ODA) at TM Forum
[oda-openapis]	**Open APIs** URL: https://www.tmforum.org/oda/implementation/open-apis/ Homepage of TM Forum open APIs
[owl2-overview]	W3C OWL Working Group. **OWL 2 Web Ontology Language Document Overview (Second Edition).** W3C Recommendation. 11 December 2012. URL: https://www.w3.org/TR/owl2-overview/ The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. This document serves as an introduction to OWL 2 and the various other OWL 2 documents. It describes the syntaxes for OWL 2, the different kinds of semantics, the available profiles (sub-languages), and the relationship between OWL 1 and OWL 2.
[owl2-primer]	P. Hitzler; M. Krötzsch; B. Parsia; P. F. Patel-Schneider; S. Rudolph. **OWL 2 Web Ontology Language Primer (Second Edition).** W3C Recommendation. 11 December 2012. URL: https://www.w3.org/TR/owl2-primer/

	The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents.
[owltime]	S. Cox; C. Little. **Time Ontology in OWL.** W3C Candidate Recommendation Draft. 15 November 2022. URL: https://www.w3.org/TR/owl-time/ OWL-Time is an OWL-2 DL ontology of temporal concepts, for describing the temporal properties of resources in the world or described in Web pages. The ontology provides a vocabulary for expressing facts about topological (ordering) relations among instants and intervals, together with information about durations, and about temporal position including date-time information.
[rdf11-concepts]	R. Cyganiak; D. Wood; M. Lanthaler. **RDF 1.1 Concepts and Abstract Syntax.** W3C Recommendation. 25 February 2014. URL: https://www.w3.org/TR/rdf11-concepts/ The Resource Description Framework (RDF) is a framework for representing information in the Web. This document defines an abstract syntax (a data model) which serves to link all RDF-based languages and specifications.
[rdf12-concepts]	O. Hartig; P.-A. Champin; G. Kellogg; A. Seaborne. **RDF 1.2 Concepts and Abstract Syntax.** W3C Working Draft. 22 August 2024. URL: https://www.w3.org/TR/rdf12-concepts/ The Resource Description Framework (RDF) is a framework for representing information in the Web. This document defines an abstract syntax (a data model) which serves to link all RDF-based languages and specifications. RDF 1.2 introduces triple terms as another kind of RDF term.
[rdf11-mt]	P. J. Hayes; P. F. Patel-Schneider. **RDF 1.1 Semantics**. W3C Recommendation. 25 February 2014. URL: https://www.w3.org/TR/rdf11-mt/ This document describes a precise semantics for the Resource Description Framework 1.1 and RDF Schema. It defines a number of distinct entailment regimes and corresponding patterns of entailment. It is part of a suite of documents which comprise the full specification of RDF 1.1.
[rdf11-primer]	G. Schreiber; Y. Raimond. **RDF 1.1 Primer**. W3C Working Group Note. 24 June 2014. URL: https://www.w3.org/TR/rdf11-primer/ This primer is designed to provide the reader with the basic knowledge required to effectively use RDF. It introduces the basic concepts of RDF and shows concrete examples of the use of RDF
[rdf11-schema]	D. Brickley, R. V. Guha. **RDF Schema 1.1**. W3C Recommendation. 25 February 2014. URL: http://www.w3.org/TR/rdf-schema/

	RDF Schema provides a data-modelling vocabulary for RDF data. RDF Schema is an extension of the basic RDF vocabulary.
[rdf11-turtle]	D. Beckett; T. Berners-Lee; E. Prud'hommeaux; G. Carothers. **RDF 1.1 Turtle, Terse RDF Triple Language**. W3C Recommendation. 25 February 2014. URL: https://www.w3.org/TR/turtle/ This document defines a textual syntax for RDF called Turtle that allows an RDF graph to be completely written in a compact and natural text form, with abbreviations for common usage patterns and datatypes.
[rdf12-turtle]	G. Kellogg; D. Tomaszuk. **RDF 1.2 Turtle, Terse RDF Triple Language**. W3C Working Draft. 29 August 2024. URL: https://www.w3.org/TR/rdf12-turtle/ This document defines a textual syntax for RDF called Turtle that allows an RDF graph to be completely written in a compact and natural text form, with abbreviations for common usage patterns and datatypes. RDF 1.2 Turtle introduces triple terms as a fourth kind of RDF term which can be used as the object of another triple, making it possible to make statements about other statements.
[rdf11-xml-syntax]	F. Gandon; G. Schreiber. **RDF 1.1 XML Syntax**. W3C Recommendation 25 February 2014. URL: http://www.w3.org/TR/rdf-syntax-grammar/. This document defines an XML syntax for RDF called RDF/XML.
[rdf12-trig]	G. Kellogg; D. Tomaszuk. RDF 1.2. TriG, RDF Dataset Language. W3C Working Draft, 13 June 2024 URL: https://www.w3.org/TR/rdf12-trig/#dfn-trig-document This document defines a textual syntax for RDF called TriG that allows an RDF dataset to be completely written in a compact and natural text form, with abbreviations for common usage patterns and datatypes. TriG is an extension of the Turtle format.
[rfc2119]	S. Bradner. **Keywords for use in RFCs to Indicate Requirement Levels**. IETF Best Practices RFC2119. March 1997. URL: https://datatracker.ietf.org/doc/html/rfc2119 This document specifies an Internet Best Current Practices regarding words used to signify the requirements within a specification.
[rfc9315-ibn]	A. Clemm; L. Ciavaglia; L. Z. Granville; J. Tantsura. **Intent-Based Networking - Concepts and Definitions**. IRTF Informational RFC9315. October 2022 URL: https://datatracker.ietf.org/doc/html/rfc9315 Informational document that summarizes the concepts and definitions by the IRTF Network Management Research Group regarding Intent Based Networks.
[rfc9315-ic]	C. Li; O. Havel; A. Olariu; P. Martinez-Julia; J. Nobre; D. Lopez. **Intent Classification**. IRTF Informational RFC9316. October 2022. URL: https://datatracker.ietf.org/doc/rfc9316/ This document proposes an intent classification methodology and an intent taxonomy.

[sid-home]	Information Framework (SID). TM Forum Information Framework Homepage. URL: https://www.tmforum.org/oda/information-systems/information-framework-sid/ This is the homepage of the ODA Information Framework (SID).
[sparql11]	The W3C SPARQL Working Group. **SPARQL 1.1 Overview**. W3C Recommendation 21 March 2013. URL: https://www.w3.org/TR/sparql11-overview/ This document is an overview of SPARQL 1.1. It provides an introduction to a set of W3C specifications that facilitate querying and manipulating RDF graph content on the Web or in an RDF store.
[skos]	Simple Knowledge Organization System (SKOS) - Homepage URL: https://www.w3.org/2004/02/skos/ SKOS is an area of work developing specifications and standards to support the use of knowledge organization systems (KOS) such as thesauri, classification schemes, subject heading lists and taxonomies within the framework of the Semantic Web
[skos-ref]	A. Miles; S. Bechhofer eds. SKOS Simple Knowledge Organization System Reference. W3C Recommendation. 18 August 2009. URL: https://www.w3.org/TR/2009/REC-skos-reference-20090818/ This document defines the Simple Knowledge Organization System (SKOS), a common data model for sharing and linking knowledge organization systems via the Web.
[tr290a-icm]	J. Niemöller. **Intent Common Model - Intent Expression**. TM Forum TR290A. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/intent-common-model-intent-expression-v3-6-0-tr290a/ This document is part of the TR290 series specifying the intent common model as part of the TM Forum Intent Ontology (TIO). The focus of this document is the specification of how intent is structured and expressed.
[tr290b-report]	J. Niemöller. **Intent Common Model - Intent Reporting**. TM Forum TR290B. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/intent-common-model-intent-reporting-v3-6-0-tr290b/ This document is part of the TR290 series specifying the intent common model as part of the TM Forum Intent Ontology. The focus of this document is the specification of how intent is structured and expressed.
[tr291a-validity]	J. Niemöller. **Intent Validity - Intent Extension Model**. TM Forum TR291A. Version 3.6.0.URL: https://www.tmforum.org/resources/introductory-guide/tr291a-intent-validity-intent-extension-model-v3-6-0/ With the intent validity model, the intent owner can delegate the validity control to the handler by specifying the detailed conditions of validity within the intent. The model allows defining complex validity schemes involving multiple types of conditions and alternative requirements with mutually exclusive validity.

[tr291b-probe]	J. Niemöller. **Intent Probing - Intent Extension Model**. TM Forum TR291B. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr291b-intent-probing-intent-extension-model-v3-6-0/ This document specifies intent probing. Probing of intent is a dialog between an intent owner and intent handler for the purpose of feasibility and potential outcome assessment of intent requirements.
[tr291c-best]	J. Niemöller. **Proposal of Best Intent - Intent Extension Model**. TM Forum TR292E. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr291c-proposal-of-best-intent-intent-extension-model-v3-6-0/ This document specifies the intent and intent report vocabulary for proposal of the best intent procedure on an intent API.
[tr291g-preference]	J. Niemöller. **Preference of Intent Handling Outcomes - Intent Extension Model**. TM Forum TR292G. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr291g-preference-of-intent-handling-outcomes-intent-extension-model-v3-6-0/ This intent extension model specifies the vocabulary that enables an intent handler to request a statement of preference for a potential outcome from the owner of an intent. The intent handler would generate intent reports representing expected outcomes of the solution alternatives under consideration. This way it can communicate the potential outcomes to the intent owner and request a statement of preference. This model also specifies how the intent owner can convey its preferences regarding the presented alternative outcomes back to the intent handler.
[tr291h-guarantee]	D. Green. **Intent Guarantee - Intent Extension Model**. TM Forum TR292H. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr291h-intent-guarantee-intent-extension-model-v3-6-0/ The Intent Guarantee model extends the vocabulary of the TM Forum Intent Ontology (TIO) to support the request for guarantees that are future looking and that are statements of enduring compliance.
[tr291i-utility]	J. Niemöller. **Utility - Intent Extension Model**. TM Forum TR292I. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr291i-utility-intent-extension-model-v3-6-0/ The utility intent extension model specified in this document introduces optional utility information to be added into an intent. Utility information specifies a function to be used to map the values of a metric into a utility score. This utility score is a measure of business value. Having this information in the intent and associated with requirements expressed with the same metrics allows an intent owner to convey its notion of business value and its preferences to the intent handler. It is doing so in a quantifiable way that is directly enabling calculations within the intent handler for making optimization and conflict resolution decisions.

[tr292a-imo]	J. Niemöller. **Intent Management Elements**. TM Forum TR292A. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr292a-intent-management-elements-v3-6-0/ The intent management elements model is an ontology within the TM Forum intent ontology. It introduces basic terminology used to describe intent based operation. The TM Forum Autonomous Networks Project introduces concepts around the use of intent within an autonomous network. For example, it introduces an architecture and functions to participate in intent based operation, roles within intent life cycle management and the intent API.
[tr292b-imo]	J. Niemöller. **Intent Management State Machines**. TM Forum TR292B. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr292b-intent-management-state-machines-v3-6-0/ The intent management state machines model is an ontology within the TM Forum intent ontology. It introduces a generic state machine describing the status of intent handling and the compliance of the system to the requirements expressed by an intent. The intent management state machines model introduces a standard vocabulary to be used between the intent handler and intent owner to communicate the intent handling status and the stage in the intent life cycle. The model defines a generic state machine of intent handling. States reflect compliance to the intent requirements and stages in the intent life cycle. State transitions serve as events that indicate that a significant change in the intent handling status has occurred. A separate, but associate state machine is defined for communicating the progress of intent updates.
[tr292c-fun]	J. Niemöller. **Function Definition Ontology**. TM Forum TR292C. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr292c-function-definition-ontology-v3-6-0/ This document introduces functions as elements in intent expression. They improve intuition when working with the intent models. Functions are a versatile concept. They can be used, for example, for defining logical or numerical operators or to define the creation and modification of data structure. This document provides an ontology and vocabulary for definition and formal description of functions within the TM Forum Intent Ontology (TIO).
[tr292d-quan]	J. Niemöller. **Quantity Ontology**. TM Forum TR292D. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr292d-quantity-ontology-v3-6-0/ This document defines quantities and a set of operator functions that allow us to define, modify, combine and compare quantity objects. Defining goals based on KPIs and metrics is a common task in intent expression. They often consist of a numerical value in combination with a unit. Quantities, as defined in this document are exactly that: a class with properties representing a numerical literal in combination with a unit expression. This document also defines operations for constructing, modifying and combining quantities. Further operators allow comparing and derive inference from quantities.

[tr292e-log]	J. Niemöller. **Conditions and Logical Operators**. TM Forum TR292E. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr292e-logical-operators-v3-6-0/ The ontology model specified in this document defines introduces a set of logical operators available for expressing logical relationships in intent modeling.
[tr292f-set]	J. Niemöller. **Set Operators**. TM Forum TR292F. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr292f-set-operators-v3-6-0/ This document defines an ontology model of set operators. Requirement specification in intents often requires formulating conditions based on sets of things. For example, the state of a network resource shall be one of a set of “good” states in order for the system to meet the requirement. Another example is the requirement that a resource must not be allocated within certain distinct regions or sites. All this involves sets and logic regarding intersection of sets and membership of individuals within a set.
[tr292g-met]	J. Niemöller. **Metrics and Observations**. TM Forum TR292G. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr292g-metrics-and-observations-v3-6-0/ This document defines an ontology model for defining and using metrics and managing observations, such as measurements concerning the metrics.
[tr292h-mf]	J. Niemöller. **Mathematical Functions Definition and Collection**. TM Forum TR292H. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr292h-mathematical-functions-definition-and-collection-v3-6-0/ This document defines a set of mathematical functions and vocabulary to manage value ranges and composite functions.
[tr292i-sec]	L. Abdelrazek. **Security Ontology**. TM Forum TR292E. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr292i-security-ontology-v3-6-0/ The ontology model defined in this document defines the necessary vocabulary that can be used in the security domain. The focus of this document is the specification of how a security intent is expressed.
[tr293-connector]	M. R. Raza. **Connector Model**. TM Forum TR293. Version 2.0.0. URL: https://www.tmforum.org/resources/technical-report/tr293-connector-model-v2-0-0/ This document describes the vocabulary for creating connector models. A connector model introduces a proxy IRI for artifacts from other models that use different identification schemes.
[tr299-specification]	J. Niemöller. **Intent Specification**. TM Forum TR299. Version 3.6.0. URL: https://www.tmforum.org/resources/introductory-guide/tr299-intent-specification-v3-6-0/

	This document specifies the vocabulary for expressing intent specifications. An intent specification defines rules and constraints regarding allowed content of intent. This allows generating and verifying well-formed intents. Many factors can be considered, such as the nature of an autonomous domain, the services and resources exposed by it and capabilities and properties of intent handlers. Intent specifications can be associated with product and service specifications within catalogs.
[uml2]	**Unified Modeling Language Specification, Version 2.0**. Object Management Group. July 2005. URL: https://www.omg.org/spec/UML/2.0/ The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system.
[unicode]	**The Unicode Standard**URL: http://www.unicode.org/versions/latest/ This page presents an overview of the latest version of the Unicode standard with links to all its distinct specifications.
[uri]	T. Berners-Lee; R. Fielding; L. Masinter. **Uniform Resource Identifier (URI): Generic Syntax**. IETF Internet Standard RFC3986. January 2005. URL: https://datatracker.ietf.org/doc/html/rfc3986 A Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or physical resource. This specification defines the generic URI syntax and a process for resolving URI references that might be in relative form, along with guidelines and security considerations for the use of URIs on the Internet.
[w3c]	**World Wide Web Consortium (W3C)**. URL: https://www.w3.org/ Homepage of the World Wide Web Consortium (W3C)
[w3c-standards]	**W3C Standards and drafts** URL: https://www.w3.org/TR/?tag=data Collection of W3C published standards.
[xml10]	T. Bray; J. Paoli; M. Sperberg-McQueen; E. Maler; F. Yergeau et al. **Extensible Markup Language (XML) 1.0 (Fifth Edition)**. W3C Recommendation 26 November 2008. URL: http://www.w3.org/TR/xml This document specifies the Extensible Markup Language (XML). It is a markup language and file format for storing, transmitting, and reconstructing arbitrary data. It defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
[xml11]	T. Bray; J. Paoli; C. M. Sperberg-McQueen; E. Maler; F. Yergeau; J. Cowan. **Extensible Markup Language (XML) 1.1 (Second Edition)**. W3C Recommendation 16 August 2006, edited in place 29 September 2006. URL: https://www.w3.org/TR/xml11/

	This document specifies the Extensible Markup Language (XML). It is a markup language and file format for storing, transmitting, and reconstructing arbitrary data. It defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
[xml-names]	T. Bray; D. Hollander; A. Layman; R. Tobin. **Namespaces in XML 1.1 (Second Edition)**. W3C Recommendation 16 August 2006. URL: https://www.w3.org/TR/xml-names11/ This document specifies the use of namespaces in XML. XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by IRI references.
[xsd-0primer]	D. C. Fallside; P. Walmsley. **XML Schema Part 0: Primer Second Edition**. W3C Recommendation 28 October 2004. World Wide Web Consortium. URL: http://www.w3.org/TR/xmlschema-0/ XML Schema Part 0: Primer is a non-normative document intended to provide an easily readable description of the XML Schema facilities, and is oriented towards quickly understanding how to create schemas using the XML Schema language.
[xsd11-structures]	H. S. Thompson; D. Beech; M. Maloney; N. Mendelsohn. **W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures**. W3C Recommendation 05 April 2012. URL: https://www.w3.org/TR/xmlschema11-1/ This document specifies the XML Schema Definition Language, which offers facilities for describing the structure and constraining the contents of XML documents.
[xsd11-datatypes]	D. Peterson; S. Gao; A. Malhotra; C. M. Sperberg-McQueen; H. S. Thompson. **W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes**. W3C Recommendation 5 April 2012. URL: https://www.w3.org/TR/xmlschema11-2/ XML Schema: Datatypes is part 2 of the specification of the XML Schema language. It defines facilities for defining datatypes to be used in XML Schemas as well as other XML specifications. The datatype language, which is itself represented in XML, provides a superset of the capabilities found in XML document type definitions (DTDs) for specifying datatypes on elements and attributes.
[yaml12]	O. Ben-Kiki; C. Evans; I. döt Net. **YAML Ain't Markup Language (YAML) Version1.2**. 1 October 2009 URL: http://yaml.org/spec/1.2/spec.html This is the YAML specification the defines the YAML language. YAML is a human-friendly, cross language, Unicode based data serialization language designed around the common native data types of dynamic programming languages. It is broadly useful for programming needs ranging from configuration files to internet messaging to object persistence to data auditing and visualization.
[yaml-ld-report]	JSON-LD Community. **YAML-LD**. Final Community Report. 06 December 2023.

	URL: https://www.w3.org/community/reports/json-ld/CG-FINAL-yaml-ld-20231206/ This document defines YAML-LD, a set of conventions built on top of YAML, which outlines how to serialize Linked Data as YAML based on JSON-LD syntax, semantics, and APIs. The emergence of YAML as a more concise format for representing information previously serialized as JSON, including Linked Data, has led to the development of YAML-LD.

Reference	Description	Source	Summary
[oda]	TM Forum Open Digital Architecture	https://www.tmforum.org/oda/	

11.Administrative Appendix

11.1. Document History

11.1.1. Version History

Version Number	Date Modified	Modified by:	Description of changes
1.0.0	03-Sep-2024	Alan Pope	Final edits prior to publication

11.1.2. Release History

Release Status	Date Modified	Modified by:	Description of changes
Pre-production	03-Sep-2024	Alan Pope	Initial Release
Pre-production	14-Oct-2024	Adrienne Walcott	Updated to Member Evaluated status

11.2. Acknowledgments

This document was prepared by the members of the TM Forum Autonomous Networks team.

Team Member (@mention)	Company	Role*
Jörg Niemoller	Ericsson	Project Co-Chair, Author, Editor
Alan Pope	TM Forum	Reviewer

**Select from: Project Chair, Project Co-Chair, Author, Editor, Key Contributor, Additional Input, Reviewer*