TM Forum Specification

Intent Management API Profile

TMF921A
Maturity Level: General availability
Team Approval Date: 31-Mar-2022

Release Status: Pre-production	**Approval Status: Team Approved**
Version: 1.1.0	**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

List of Tables

1 Executive Summary

This document concerns the API Component Suite for **Intent Management** and defines the set of operations that should be offered in manage intent and intent-driven interactions in a consistent manner.

2 Introduction

2.1 Audience

This guide describes the requirements for a new *intent interface* that has usage potential throughout the Autonomous Networks Reference Architecture (IG1251). This guide is primarily focused on capturing the requirements from the collaboration projects interested in consuming and developing this intent API in a format that can then be consumed by the experts in the API Project team so that it may progress to a Stage 3 Level API specification in the TM Forum Open API program.

2.2 Concepts and Background

2.2.1 Intent Definition

First, some brief context. What is an *Intent*? TM Forum defines it as follows:

> ***"Intent is the formal specification of all expectations including requirements, goals, and constraints given to a technical system"***
>
> **IG1230 AN Technical Architecture (TM Forum, 2020)**

So Intent is saying what you expect, what you want and NOT how to do it, or even base some subtle hints at maybe how best it might be done. Just the "what"! This 'what' is referred to as the expression of the intent and part as an entity you can think of an Intent as a grammar or syntax to express your expectations.

Intent is about communication. Between multiple parties. The *Owner* and the *Handler.* Both Humans and Machines can play these *roles.* For the purposes of this API, consider the owner and handler party roles as being only Machines i.e., autonomous systems i.e., the right hand side of Figure 1-1.

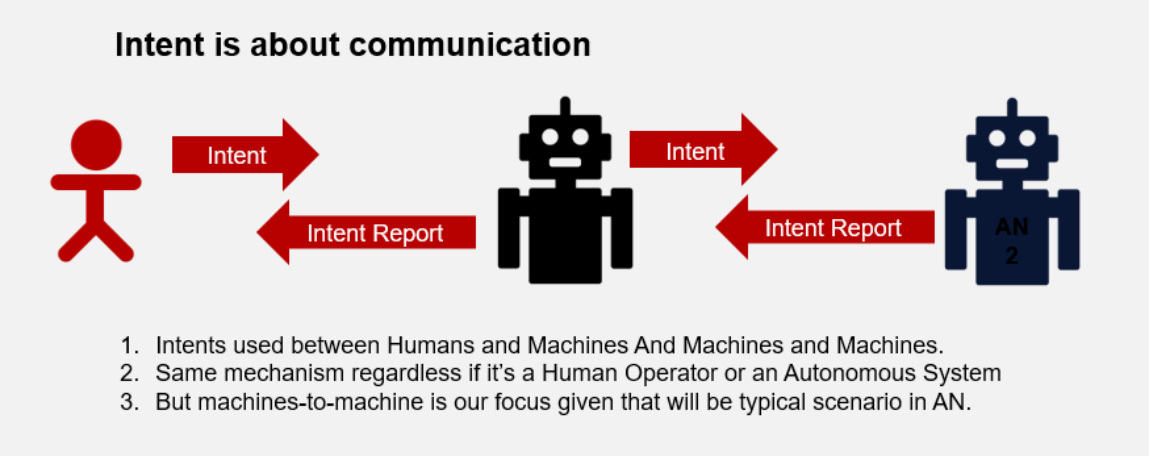


Figure 1-1 Intent is about communicating your expectations

2.2.2 Autonomous Operations requires a declarative approach

Intent is about setting expectations and goals. The concept of intent has recently shifted from a policy-centered view to one more focused on operational goals. This is also the direction taken by the TM Forum as part of the Autonomous Networks collaboration project. Intents are communicated between parties or systems. Intent expressions are the communication mechanism and this mechanism works both between humans and machines and between machines and other machines (by 'machine' here we mean an autonomous system). Communication between machines and machines is important because this is where most of the intents will be generated in an Autonomous Network.

The theory of intent has been developed some time ago, but the practical side of intent requires that we can apply the concept to everyday telco operations. Unfortunately, today's telco operations are essentially manual. The high-level strategies of the CSPs, their business priorities, etc. are only captured in documents and understood in the "heads" of the management team and then passed on to the human operations team. Decision-making is done exclusively by humans.

However, there are areas of automation that are driven by policies that in turn drive top-down decision-making. When policies do not match unexpected real-world situations, humans must step in to fill the gap and manage the unexpected. Human operations teams make the decisions that streamline these brittle processes, as shown in middle part of Figure 1-2. (Note that it is not only unplanned and unexpected situations that arise, but also *deliberate* variations and the automation systems must accommodate these variations in a non-brittle way by understanding that these variations are intentional.)

Autonomous operations go far beyond automation to change the nature of operations and give much more autonomy to the machines themselves using the intent-driven approach. Strategic intents and behavioral intents drive closed-loop processes in which humans are no longer just "in the loop" but "on the loop," meaning they are no longer just a manual step in the process but now oversee the steps taken by the machine. The big difference in terms of autonomous operations is that intent decouples the "what" from the "how," giving systems and networks that use AI the freedom to find better solutions that humans would not normally find. An intent-driven approach fundamentally enables autonomy, which means operations are faster, better, more consistent, and smarter.

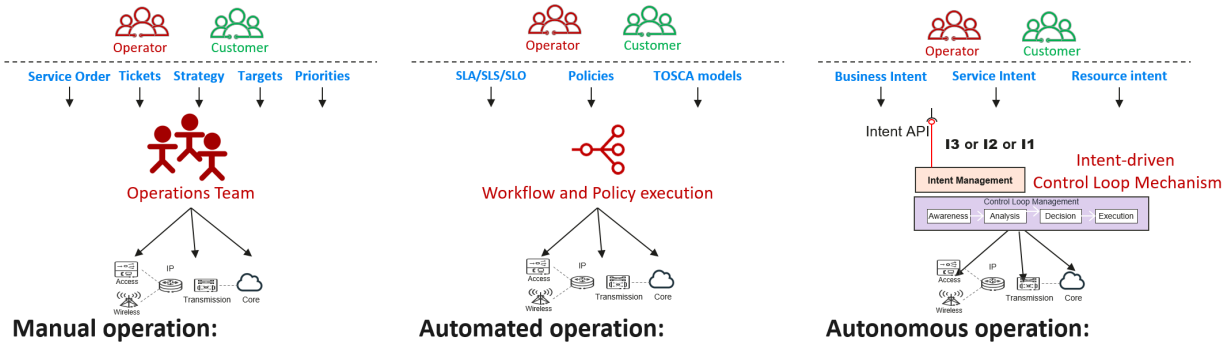


Figure 1-2 Autonomous Operations

2.2.3 Autonomous Networks Architecture

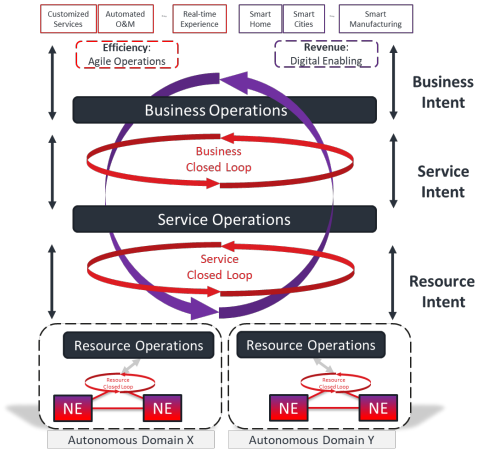
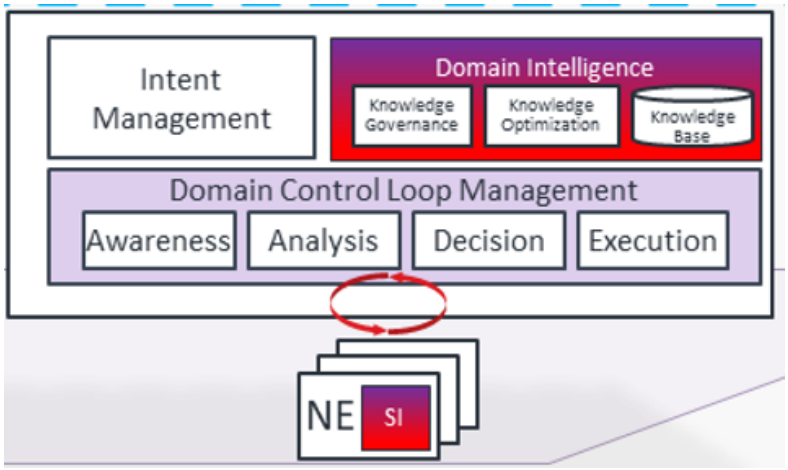


Figure 1-3 Autonomous Networks (IG1230)

The technical architecture shows how different autonomous domains can be built up to assemble the overall autonomous network and the Intent API is a key part of making this integration fabric of autonomous systems work together. The intent management function that 'houses' the Intent API endpoints.

As outlined in the TM Forum's **Autonomous Networks Reference Architecture (IG1251)**, the need for intent-driven interfaces that can be used at all operational layers is imperative to achieve the decoupled and autonomous functional blocks, termed autonomous domains. This guide describes the requirements for an Intent Management API suite that fully implements the unified intent modelling and allows for dialogue or conversational level interactions using intents.



This component suite identifies the operations that provide the functionality to allow Communications Service Providers to manage *intents* and to have interactions and dialogs using Intents.

IG1253C Intent Life Cycle Management and Interface describes the lifecycle management and interfaces required to manage intents in terms of mechanism and sequence flows. This guide provides a more generalized statement of the requirements for such interfaces. IG1253C also outlines the scope of the component suite described in this guide.

The concepts of intent and intent-driven operations are more thoroughly covered in **IG1253 Intent in Autonomous Networks**. Intents are defined as knowledge objects. As such they have a defined life cycle that needs to be actively managed. Intent management functions implement this management task. This process involves communication between intent management function *to exchange intents and intent reports*. This interface allows sending the intent, reporting on handling success, modify the intent and ultimately removing the intent. Optionally the interface allows collaborative prioritization of solutions for fulfilling the

intent. It also allows a feasibility investigation if an intent can be fulfilled and a negotiation about what level of requirements and constraints would be acceptable.

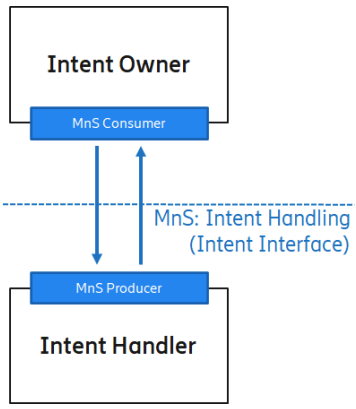


Figure 1. Intent Management Service constituting the intent interface

The main capabilities for the API are:

- Setting the intent by the intent owner
- Reporting on intent by the intent handler
- Negotiating an intent between the Owner and the Handler

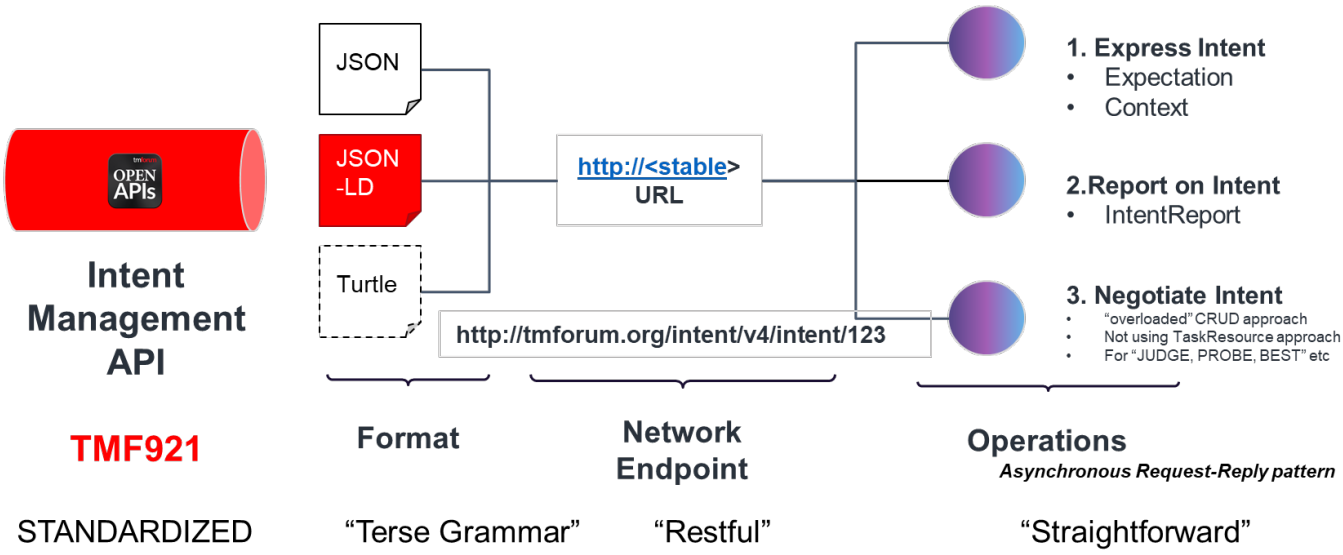


Figure 1-4 Basic Functions of Intent API and Possible Payload Formats

Currently, the team plans to develop a JSON format for the API with a JSON-LD approach for the intent expression.

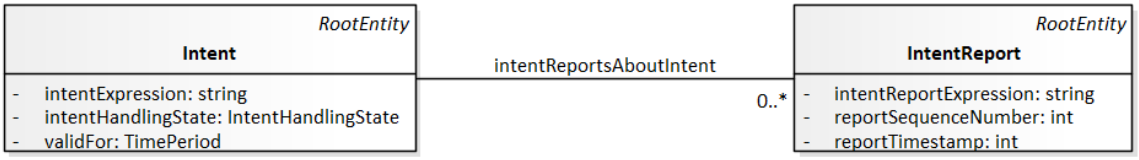


Figure 1-5 Intent Aggregate Business Entity (ABE)

This class diagram above has been agreed with SID (Frameworx) team.

3 Requirements and Use Cases

This section describes the functions that are needed to manage intent in an intent-based system.

As described in the introduction, the intent objects have their own life cycle which is managed by the intent owner and the intent handler: the intent owner that will create the intent object and its expectations (requirements, goals and constraints) and the intent handler which will consider the expectations of the intent and adapt them to the specific domain and infrastructure it is responsible for. The intent handler is also responsible for keeping the intent owner updated of the status of the intent via the intent reports.

3.1 Requirements

The Intent API suite includes functions are organized into the following areas:

- Setting the Intent
- Reporting on Intent
- Negotiating an Intent
- Profile Handling

These areas are discussed below, with the exception of those marked TBA which will be addressed in future revisions of this work.

3.1.1 Intent Setting

The Intent Setting functional area includes all activities needed from the intent owner to manage the intents which are needed from the intent handler. This includes full lifecycle support for all intent objects and contained elements such as expectation and context. There are three main areas:

- Managing Intents: functions to create, modify or delete intent and to retrieve intent objects information
- Managing Expectations: expectations can be added, updated or removed from existing intent objects and information regarding specific expectations can be retrieved from the system

- Managing Contexts: context can be added, updated or removed from existing intent or expectation objects and information regarding specific context can be retrieved from the system

Using variables as the expectation targets
An important requirement when setting the intent expectation will be to use variables in the intent definition.

Expectation target properties refer to the resources the requirement, goal or constraint is about and in many practical cases the exact resource instance to be used is a choice of the intent handler. This means the targeted instance cannot be known by the intent owner when formulating the intent. The intent owner would use variables as placeholders instead.

3.1.2 Intent Reporting

Intent reports are created and sent to the intent owner according to reporting criteria specified in reporting expectation by the intent owner. This means the intent owner can configure within the intent when and under which conditions is wants to be informed by the handler with an intent report. Typically reports are sent at major events in the intent life-cycle, such as acceptance of the intent or a modification, violation of the intent, success of fulfilling an intent, etc. Furthermore, the intent owner can configure reporting to send regular reports. This means reporting is a 'push' mechanism from the intent handler to the intent owner according to criteria set by the owner. As the intent reporting conditions are specified within the intent as additional expectation, the intent owner can change the reporting for the intent by updating it.

Target properties, which were defined with placeholders in the intent, would be substituted by references to the resource individuals that were chosen to instantiate the requirements, as soon as the handler has determined this information

3.1.3 Intent Negotiation

Intent negotiation refers to communication between the intent owner and handler regarding the feasibility of requirements or preference of solution and action outcomes. The intent API defines a set of optional procedures to provide these capabilities.

3.1.4 Intent Manager Capability Profile

The intent manager capability profile refers to an intent manager registry service. This service acts as an inventory of all available intent management function. Intent managers announce their capabilities and responsibilities through this service making them discoverable by other intent managers. This mechanism allows to determine what intents can be used and what requirement details they can contain. The intent registry service and its related interface procedures are described in IG1253D.

3.2 User Stories

Many stakeholders, both inside and outside of CSP, are potential *Intent Owner* roles. These stakeholders or actors could be, for example, a **customer** passing their customer intents to

the CSP. A **regulator** may express constraints as intents to all CSPs, and again they would be taking the role of an Intent owner. Whether the regulator is a person or a system doesn't change the fact that the role they are playing is that of an intent owner.

Other concrete roles could include the following, but the list is endless.

- Business Manager
- ML Engineer/Data Scientist
- Engineering/Operations

With this in mind, we will use the roles of Owner and Handler to describe the actors' involvement in the use case.

UC #	User Stories	Comments
UC.1	As an Intent Owner, I want to set (express) my expectations as part of an intent request (So That) Without the need to express how those expectations are to be met or how the outcome is to be reported on.	Create a new intent
UC.2	As an Intent Owner I want to modify my expectations as part of intent request	
UC.2	As an Intent Owner I want to be able to remove my intent expectations	
	As an Intent Owner I need to be able to uniquely identify each Intent and associated Intent Reports for that Intent	
	As an Intent Handler I need to report to the Intent Owner	

Figure 2-1 User stories

3.3 Illustrative Use Cases

3.3.1 Intent-driven Autonomous Networks for Smart Mobility (Proof-of-Concept project)

Figure 2-2 shows how the Intent-driven Autonomous Networks for Smart Mobility (IDAN4SM) catalyst leverages the conceptual AN framework for its smart mobility use cases. The catalyst CSP champions wanted to apply autonomous network techniques and principles to the problem space of toll road operations/transportation. This approach of Intent-driven management and operations where customer requirements are expressed as expectations (i.e., ask for what you expect to happen as an outcome, do not mandate how the outcome is achieved). The underlying systems and network will work out the how and will adapt to changing customer needs, environmental conditions, etc. By specifying customers intent (categorized by TM Forum as a business intent) we wish to decompose these into service intents and resource intents in the lower operational layers.

Each autonomous domain of the overall system architecture leverages an intent approach and this, in turn, changes the focus to each system to more self-contained, self-centered systems that focus on their respective closed loops.

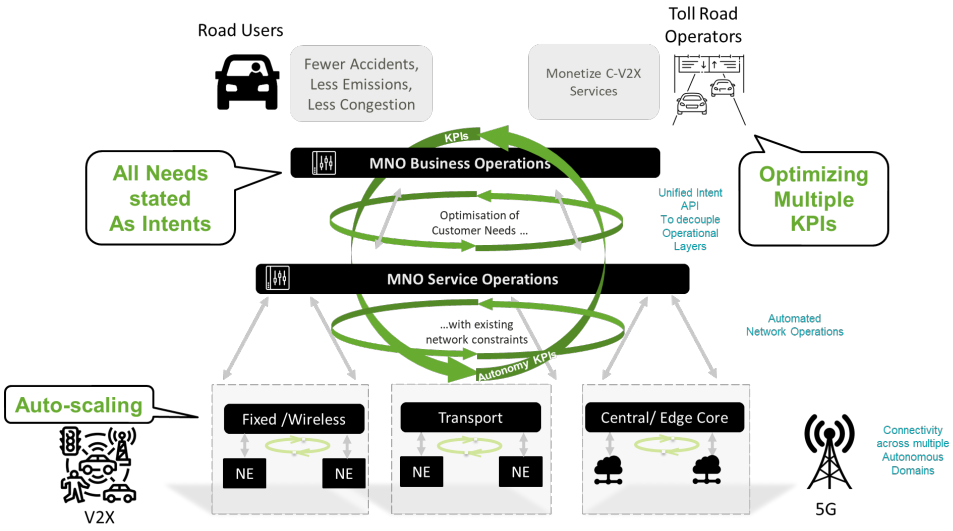


Figure 2-2 Catalyst Architecture using TM Forum AN Framework

3.3.1.1 Understanding Stakeholder requirements and intents

Table 1 Stakeholder Requirements Matrix

	Toll Road Operator (TRO) Vantage Point		Communications Service Provider (CSP) Vantage Point		
	User perspective (Road User → TRO)	**TRO Business perspective**	**CSP Customer perspective (TRO→ CSP)**	**ICT service perspective (CSP)**	**ICT resource perspective (CSP) (domain-centric)**
Requirements	- travel from source to destination as quickly, comfortably, and safely as possible - Frictionless use of toll roads, simple one-time setup, cashless, secure payment, no human contact needed, email receipt if required	- Safely optimize traffic speed, flow, and density through the toll road infrastructure - Maximize revenue whilst observing safety and customer user experience	- Provide targeted connectivity solution to enable V2X communications over the toll road infrastructure -Provide dynamic and scalable service based on road traffic patterns	- Provide resilient connectivity solution to enable V2X communications - Provide dynamic service scaling capability based on road traffic patterns	- Provide resilient connectivity solution to enable V2X communications
Metrics	- Account setup time - Account setup success rate - Average toll road journey time - Average toll price-Customer satisfaction - #Customer complaints	*Metrics (per road, or lane) - Average traffic speed - Average traffic density - Toll road revenue - Toll road traffic incident rate (per 1M km) - #Staff	- V2X Service Creation Time - V2X Service Accessibility - V2X Service Reliability - V2X Service Latency	- V2X bearer Accessibility - V2X bearer Reliability - V2X bearer Service Latency - Positioning Accuracy	Metrics per RAN/Core/Transport (not included as too detailed for this whitepaper scope)
Business Capabilities	- Single touch data entry onboarding - Zero touch transactions - E-receipt - Self-service support	Customer authentication, authorization , Customer information management, Customer interaction management, Customer lifecycle management, Customer loyalty management, Partner management Asset lifecycle management Product lifecycle management	Customer authentication & authorization, Customer information management , Customer interaction management, Customer lifecycle management, Customer loyalty management, Partner management, Asset lifecycle management, Product lifecycle management	- Order management - Partner management - SLA management - Trouble & Incident management	- Resource activation - Network performance management - Network compliance management - Network risk management - Network allocation management

3.3.1.2 Intent examples for V2X use cases

The different stakeholders that participate in the V2x use cases (e.g., speed harmonization), may translate both their business needs and technical needs to a chain of Intents. Figure 2-3 shows a typical chain of such Intents and their inter-dependencies. Each one of the roles: the Road user, the Toll Road Operator, the V2X operator, and the Digital Services Provider, may have both business Intents and Technical Intents. In some cases, the technical Intents may have multiple levels (e.g., a service intent and a slice intent). This is not necessarily a linear chain. For example, the technical service intent of the V2X operator may be derived from both its business intent and from the technical service intent of its consumer the TRO.

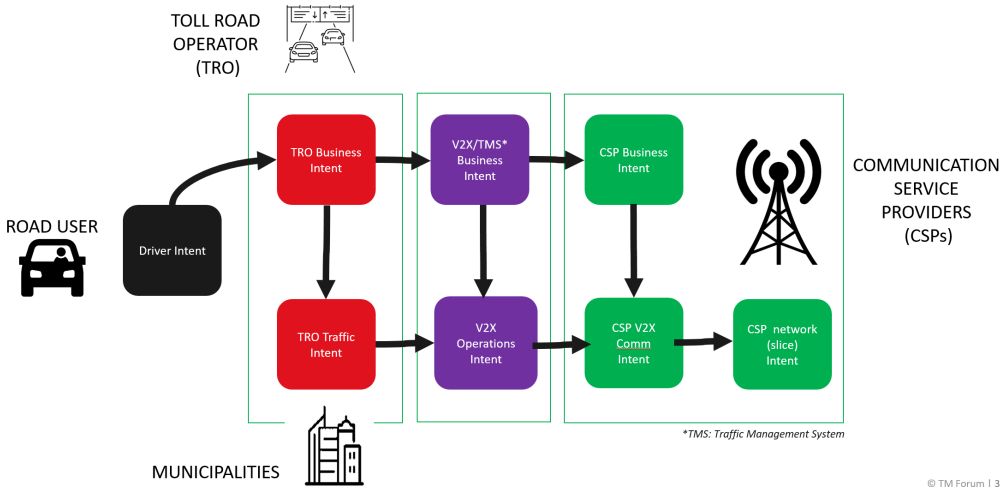


Figure 2-3 Intent Map

When looking at the details of the Intents at the level of specific targets we can see how these are being elaborated as we traverse across the Intent chains. The general concepts of business agility, safety, excellence in customer experience and being environment friendly are realized by more specific technical terms, such as latency, speed, availability of communication and throughput.

Figure 2-4 contains an overview of the eight intent examples showing text of the different expectations or objectives. This remaining of this section describes the specific requirements expressed as multiple independent intents. These Intent Expressions are names per stakeholder and listed as *bulleted expectations* in the following sub-sections.

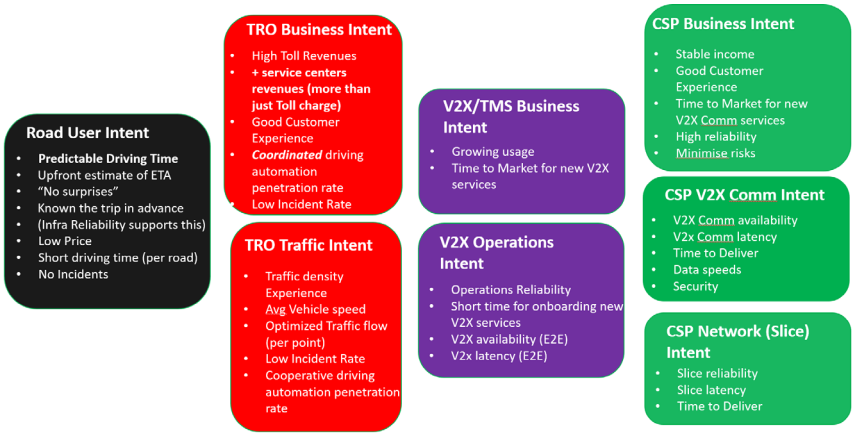
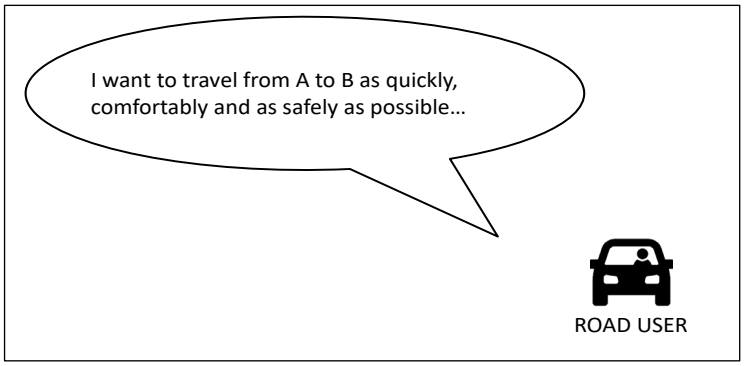


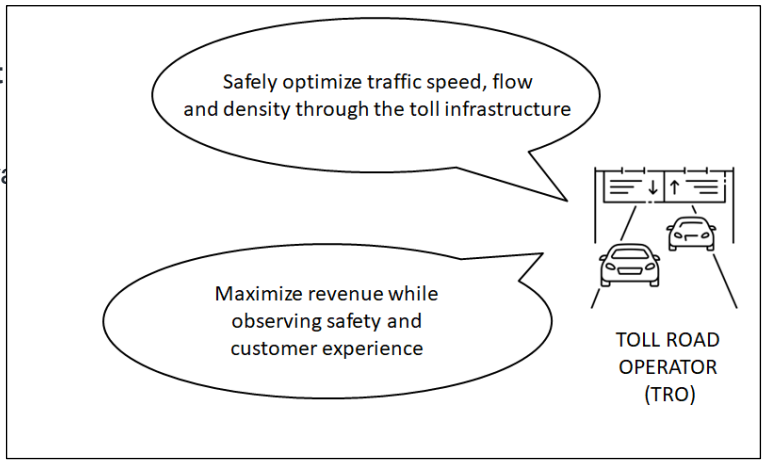
Figure 2-4 Intent Examples (complete Intent chain)

3.3.1.3 Road User Intent



- **Predictable Driving Time**
- Upfront estimate of ETA
- "No surprises"
- Known the trip in advance
- (Infra Reliability supports this)
- Low Price
- Short driving time (per road)
- No Incidents

3.3.1.4 TRO Business Intent



- High Toll Revenues
- **+ service centers revenues (more than just Toll charge)**
- Good Customer Experience
- ***Coordinated*** driving automation penetration ra
- Low Incident Rate

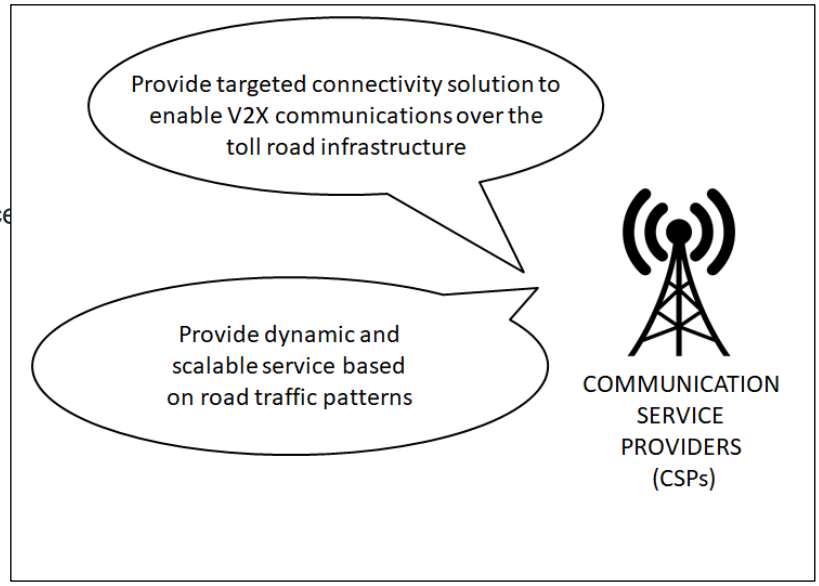
3.3.1.5 TRO Traffic Intent

- Traffic density Experience
- Avg Vehicle speed
- Optimized Traffic flow (per point)
- Low Incident Rate
- Cooperative driving automation penetration rate

3.3.1.6 V2X/TMS Business Intent

- Growing usage
- Time to Market for new V2X services

3.3.1.7 V2X Operations Intent



- Operations Reliability
- Short time for onboarding new V2X service
- V2X availability (E2E)
- V2x latency (E2E)

3.3.1.8 CSP Business Intent

- Stable income
- Good Customer Experience
- Time to Market for new V2X Comm services
- High reliability
- Minimize risks

3.3.1.9 CSP V2X Comm Intent

- V2X Comm availability
- V2x Comm latency
- Time to Deliver
- Data speeds
- Security

3.3.1.10 **CSP Network Slice Intent**

- Slice reliability
- Slice latency
- Time to Deliver

3.3.1.11 **Network Slice Intent Examples**

The following 3 example individual *service intents* describe the requirements for network slices in 3 discrete scenarios.

Example #1 – Throughput assuming a minimal volume

1. RAN UE Throughput (or total upstream throughput) > X if Slice Volume > Y (to ensure at least a minimal level of activity)

Example #2 – Guaranteed # of supported users for different busy time levels

1. #Registered Subscribers of Network Slice Instance > X during week days
2. #Registered Subscribers of Network Slice Instance > y during Weekends
3. #Registered Subscribers of Network Slice Instance > 150% avg # of registered subscribers on week days

Example #3 – Latency for Urban /Rural areas

1. E2E Latency – Low Capacity Area (Latency < X),
2. E2E Latency – High Capacity Area (Latency < Y)

These E2E Latencies can be automated a bit by calculating distances (density) automatically.

3.4 Functions

3.5 Create a new intent

Pre-condition: An intent service instance with the same id does not exist

Post-condition:

- An intent instance has been created and is captured in the intent instance inventory of the domain.
- A response with intent id is returned
- *Aside: An intent report is sent to the owner are only sent if this event is explicIty listed in the reporting expectation.*

Depending on the intent and sometimes the details of a particular intent creation request, the acceptance or rejection of the intent will be immediately returned. In other cases, the immediate response will simply be that the intent has been received.

Error**:** If the intent is immediately rejected a rejection reason will be returned. Possible reasons include:

- Unsupported expectations: The intent contains expectation classes the intent handler does not support.
- Unsupported information model in expectation: For example, a KPI from an unsupported metrics model is used within a known expectation object.
- Out of scope: The intent defines details that are not in the domain scope of the intent handler.

Warning: The intent handler can accept the intent but raise a warning if needed. Possible reasons include:

- Unsupported Context: The intent uses a context class the intent handler does not support and has not implemented.
- Unsupported information model in context

3.6 Modify existing intent

Pre-condition: An intent instance exists (i.e., the Intent ID must be a valid ID that exists)

Post-condition:

- Intent instance has been modified and updates captured in the intent instance inventory of the domain.
- *Aside: An intent report is sent to the owner are only sent if this event is explicIty listed in the reporting expectation.*

Depending on the intent and sometimes the details of a particular intent update request, the acceptance or rejection of the intent update will be immediately returned. In other cases, the immediate response will simply be that the intent update has been received.

Error: If the intent is immediately rejected an update rejection reason will be returned, and the intent instance will be unchanged. Same errors and warning are expected as in 3.1

3.7 Remove intent

Pre-condition: An intent instance exists

Post-condition:

- The intent is removed.
- An intent report is sent to the owner

Depending on the intent and sometimes the details of a particular intent removal request, the removal of the intent instance and its removal from the intent instance inventory of the domain could be immediate. In other cases, the immediate response will simply be that the intent removal has been received.

Error: If the intent removal fails e.g., some internal dependency, then the intent instance will be unchanged

3.8 Retrieve intent

Pre-condition: Intent does exist

PostCondition: The Intent and all its components (expectations and context) are returned

Error: Request for non-existing intent or attributes will return an error

Note: Intents are immutable by the receiver. This means that a retrieval will always return exactly the same content. An intent owner should not have the need to read back its own intents as nothing can have changed. Reading intent is provided for completeness - and can be used , for example, where a third party (not owner or handler of this intent) wants to read an intent.

3.9 Intent Report Event

Pre-condition: A condition occurs the triggers a notification, such as:

- Identification of a notifiable expectation violation which requires a new Intent Report
- Identification of a situation which requires a Judge/Preference dialogue
- Status change on an Intent object
  - IntentReceived: A new intent was received from the intent owner.
  - IntentAccepted: A new intent has been accepted
  - IntentRejected: The intent was rejected.
  - IntentRemoval: The intent owner has ordered a removal of the intent
  - IntentHandlingEnded: The intent handler has finished all tasks associated with the removal of the intent
  - StateComplies: The system state changes from being degraded to compliant
  - StateDegrades: The system state changes from being compliant to degraded
  - UpdateReceived: An update for the intent was received
  - UpdateAccepted: The update was accepted and the intent handler proceeds to replacing the intent content

- UpdateRejected: The update was rejected and the intent handler continues with the previous version of the intent
- UpdateFinished: The intent handler has finised executing an successful update

Post-condition:
A notification is generated providing details an Intent Report and sent to the registered callback.

Error: If the callback id is not set an error will be generated.

Note: - The list of reporting events can be extended in the future by intent extension models (in the Intent Ontology model). This way events specific to use cases and domains can be supported as needed.

3.10 Judge Intent Notification (Escalation / Request for Approval)

This is a notification sent from handler to owner.

Pre-Condition:
- The intent handler has determined multiple potential actions and cannot decide which is better.
- A condition occurs and the intent handler is not able to decide what is better.

Post-condition:
- Each intent report represents the expected outcome of an action the intent handler can do
- One or more intent reports are sent to the registered call back for the intent owner to review and decide
- Each intent report has a unique identifier which will be used by the Intent owner to respond with its preference

Error: If the callback id is not set an error will be generated.
In the case of 'late' owner responses a timeout error make be used. This response timeout may be communicated in the judge request.

3.11 Preference Intent (Approve / Answer escalation)

Pre-condition:
- A ***Judge*** request has been sent to the Intent Owner
- The intent report(s) have unique identifiers that can be used by the owner to reply to the intent handler

Post condition:
- The intent handler will perform some adjustments following the recommendation of the intent owner

Error: If the preference could not be stated by the Intent owner e.g., the Judge request can be resent at a later point and the Preference Intent retried.

3.12 Probe Intent

Pre-condition: The intent wants to explore if a particular intent is possible for an intent handler

Post condition: The intent report is sent informing the owner whether the intent is possible or not
Error: If the requested change is not feasible e.g., resources are not available an error will be returned, and intent instance will be unchanged

3.13 Best Intent

Pre-condition: the intent owner wants to explore the “best” value that can be achieved for an specific expectation
Post condition:
Error: If the intent request is not translatable or verifiable an error will be returned.

3.14 Propose Intent

Pre-condition: the intent handler response to a “best” request from the owner
Post condition:
Error: If the intent request is not translatable or verifiable an error will be returned.

4 Component Capabilities, Flows and Sequence Diagram

4.1 Requirements to Functions

Requirement	Function
Intent	• Create new Intent • Modify existing Intent • Remove Intent • Retrieve Intent
Intent Report	• IntentReport event
Negotiation	• Judge Intent Notification (Escalation / Request for Approval) • Preference Intent (Approve / Answer escalation) • Probe Intent • Remove probe Intent • Best Intent • Propose Intent
Intent Manager Capability Profile	• FFS

Table 2. Intent Requirements, Functions and associated APIs

4.2 Sequence Diagrams

The following diagrams are not exhaustive and are examples only.

4.2.1 Create Intent

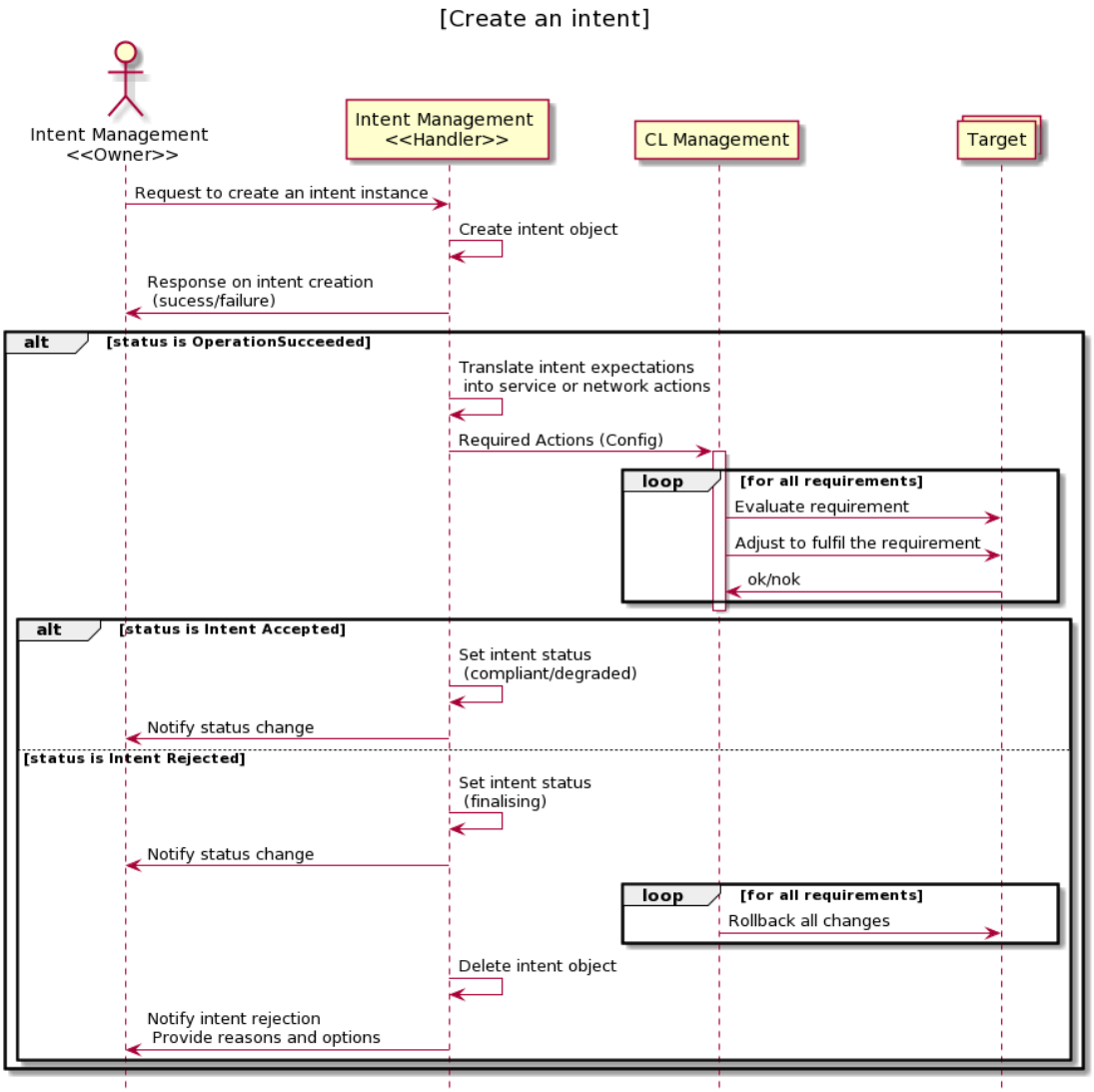


Figure 4-1. Create an Intent

4.2.2 Modify Intent

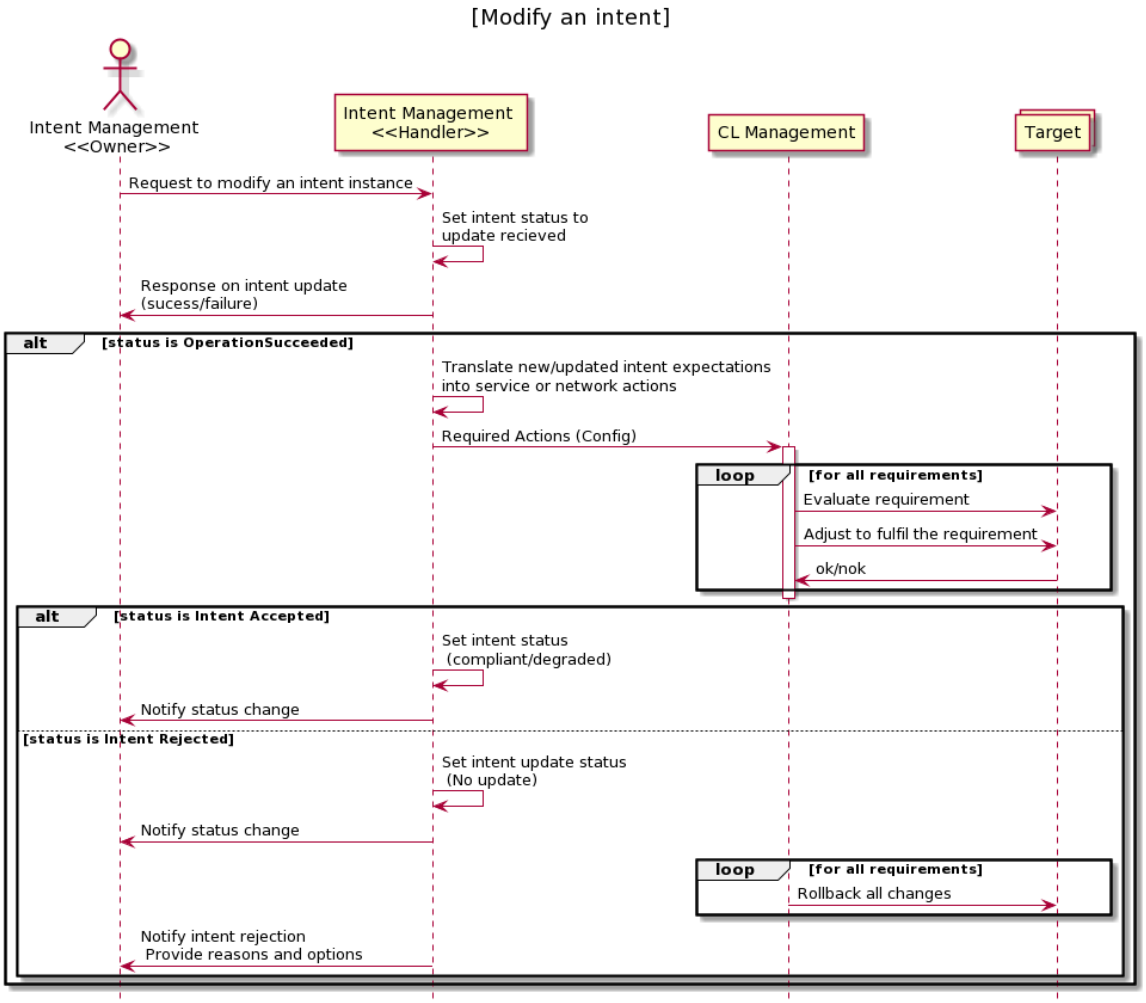


Figure 4-2. Modify an Intent

4.2.3 Remove Intent

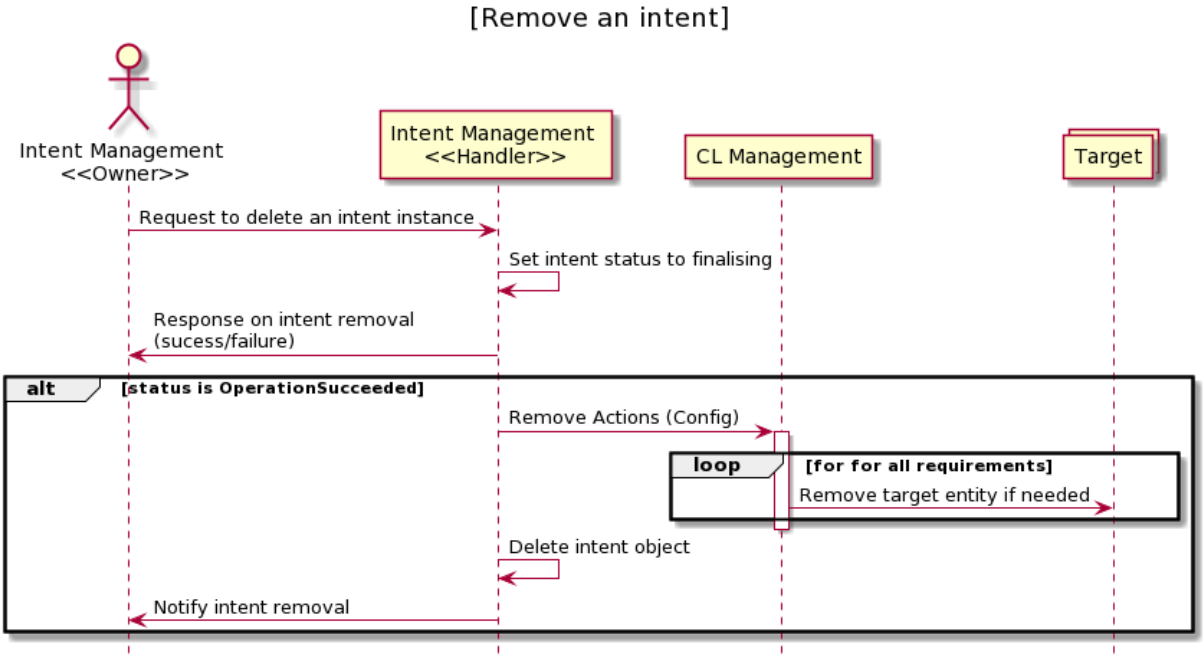


Figure 4-3. Remove an Intent

4.2.4 Judge / Preference Interaction

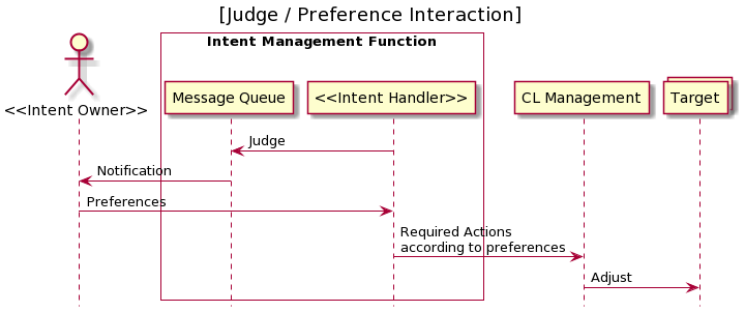


Figure 4-4. Judge / Preference Interaction

4.2.5 Probe Intent Interaction

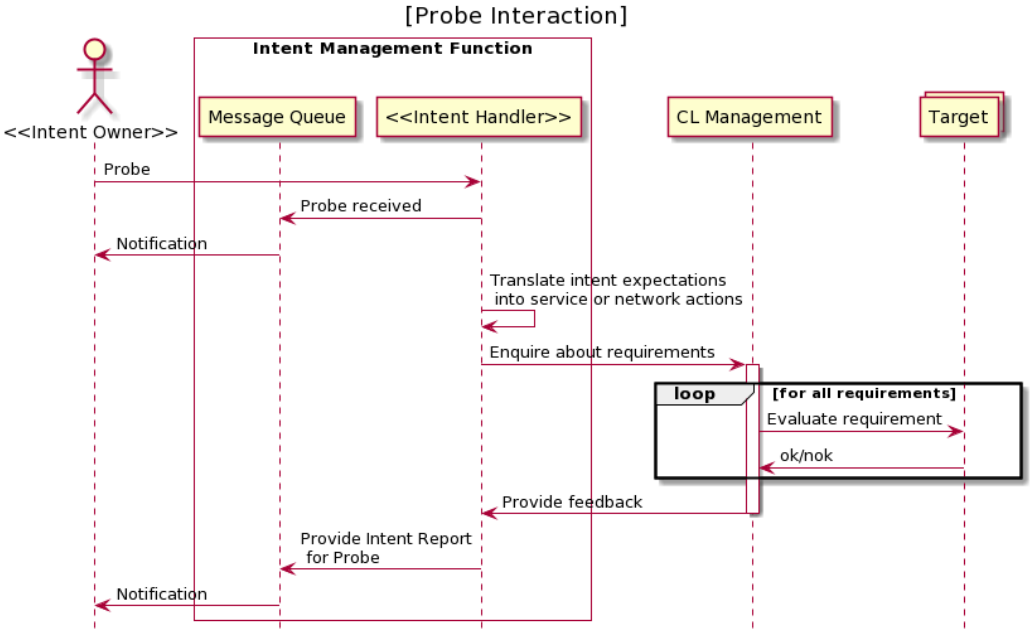


Figure 4-5. Probe Intent Interaction

4.2.6 Best / Propose Interaction

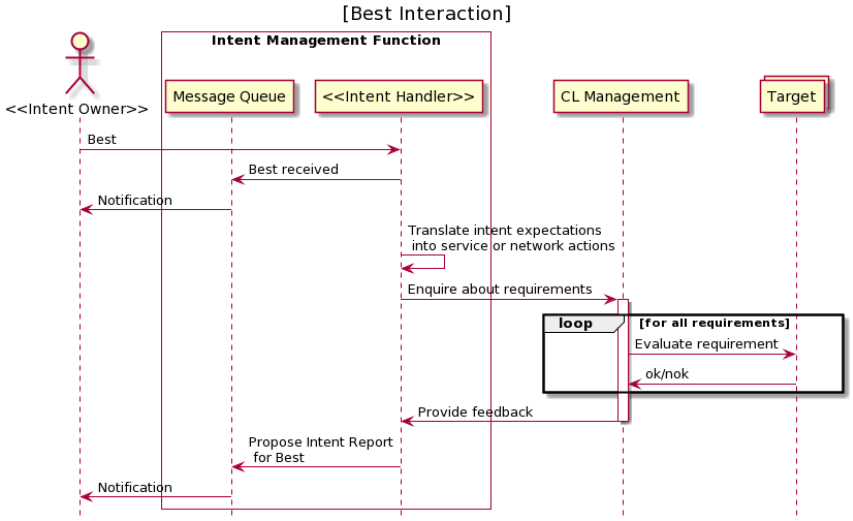


Figure 4-6. Best Intent Interaction

5 Domain Entities

- Please provide a summary of the domain entities exposed by the API.
- For each of them describe the attributes supported by the Entities.

5.1 Intent

The purpose of intent is to define and communicate knowledge about requirements, goals and constraints to a system in a way that allows automated processes to reason about it and derive suitable decisions and actions. Further details about intent can be found in **IG1253 Intent in Autonomous Networks.**

5.1.1 Entity Attributes

For example :

Name	Data type	Properties	Description
ID	String	- multiplicity is 1 - unique - mandatory	A unique identifier for an Intent.
intentExpression	String	- multiplicity is 1 - unique - mandatory	The intent expression information may include particular *objective* and possibly some related details – We call this an *Expectation* and *Context* in the Ontology model. *Example: another **intent** expression could be 'optimize the network 123 to satisfy XYZ performance requirements ‘ *Examples: the **intent** expression such as 'Car A wants to obtain V2X communication service' The structure of the string is defined in IG1253A and the Intent API will use an embedded JSON-LD to describe the content of this IntentExpression.
validFor	TimePeriod	- multiplicity is 1 - unique - mandatory	The period of time during which the intent is applicable.
lastUpdate	DateTime	- mandatory	LastUpdate attribute which should be set to the current time of the last modification

5.1.2 Associations

Describe the associations relative to that entity. The associations will be used to add endpoints to this entity to navigate the relationship.

For example:

Relationship attribute	datatype	properties	description
intentHandlingState	enum	- multiplicity is 1	RECEIVED COMPLIANT DEGRADED FINALIZING

5.2 Intent Report

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

5.2.1 Entity Attributes

For example :

Name	Data type	Properties	Description
ID	String	- multiplicity is 1 - unique - mandatory	A unique identifier for an IntentReport.
intentReportExpression	String	- multiplicity is 1 - unique - mandatory	The intent report expression is similar to the IntentExpression in Intent is that is a structured string according to Intent Ontology (the newly christened TM Forum Intent Ontology or TIO). The concept varies from intentExpression in that contains information on how the Intent Handler can handle the received intent
reportTimestamp	Timestamp	- multiplicity is 1 - unique - mandatory	The timestamp of when intentReport was created.

5.2.2 Associations

Describe the associations relative to that entity. The associations will be used to add endpoints to this entity to navigate the relationship.
For example:

Relationship attribute	datatype	properties	description
intentID	String	- multiplicity is 1 - mandatory	The Identity of the Intent that the IntentReport describes.

6 Domain Events

- Provide a description of the Event Types supported by the API.
- For each event type provide a description of the Event attributes.

6.1 IntentReceived

A new intent was received from the intent owner

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
name	String	- mandatory	Name of the intent received
description	String	- optional	

6.2 IntentAccepted

A new intent has been accepted

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
IntentID	String	- mandatory	ID of the Intent received
name	String	- mandatory	Name of the intent received
description	String	- optional	

6.3 IntentRejected

The intent was rejected

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
name	String	- mandatory	Name of the intent received
description	String	- optional	

6.4 IntentRemoval

The intent owner has ordered a removal of the intent

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event

name	datatype	properties	description
IntentID	String	- mandatory	ID of the Intent to be removed
name	String	- mandatory	Name of the intent to be
description	String	- optional	

6.5 IntentHandlingEnded

The intent handler has finished all tasks associated with the removal of the intent

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
IntentID	String	- mandatory	ID of the Intent removed
name	String	- mandatory	Name of the intent removed
description	String	- optional	

6.6 StateComplies

The intent state changes from being degraded to compliant

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
IntentID	String	- mandatory	ID of the Intent
name	String	- mandatory	Name of the intent
description	String	- optional	

6.7 StateDegrades

The intent state changes from being compliant to degraded

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
IntentID	String	- mandatory	ID of the Intent
name	String	- mandatory	Name of the intent
description	String	- optional	

6.8 UpdateReceived

An update for the intent was received

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
IntentID	String	- mandatory	ID of the Intent to be updated
name	String	- mandatory	Name of the intent to be updated
description	String	- optional	

6.9 UpdateAccepted

The update was accepted and the intent handler proceeds to replacing the intent content

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
IntentID	String	- mandatory	ID of the Intent to be updated
name	String	- mandatory	Name of the intent to be updated
description	String	- optional	

6.10 UpdateRejected

The update was rejected and the intent handler continues with the previous version of the intent

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
IntentID	String	- mandatory	ID of the Intent to be updated
name	String	- mandatory	Name of the intent to be updated
description	String	- optional	

6.11 UpdateFinished

The intent handler has finished executing an successful update

Event attributes.

name	datatype	properties	description
ID	String	- mandatory	A unique identifier for the Event
IntentID	String	- mandatory	ID of the Intent to be updated
name	String	- mandatory	Name of the intent to be updated
description	String	- optional	

7 Functions and API Mappings

{api_root} = https://...../intentManagement/v4

Function Name	Already Defined ?	Candidate for Common API?	API Operation and Notification Mapping	Comment and Constraints
Set Intent **Retrieve intent** **Intent Report Event** **Remove Intent**	N	NA	**POST {api_root}/intent** **GET {api_root}/intent/{id}** **GET {api_root}/intent/{id}** **DELETE {api_root}/intent/{id}**	...
Judge Intent Notification **Preference Intent**	N	NA	**POST {api_root}/intent/{id}** **PUT {api_root}/intentReport/{id}**	Owner calls Handler!
Probe Intent **Best Intent** **Propose Intent**	N	NA	**POST {api_root}/intent/probe=true** **POST {api_root}/intent/?best=true** **POST {api_root}/intent/**	Owner calls Handler!
Intent Manager Capability Profile	N	NA	**FFS**	

Table 3. API mappings

7.1 Notification Tables

API Name	Notifications
Intent API	• IntentReceived • IntentAccepted • IntentRejected • IntentRemoval • IntentHandlingEnded • StateComplies • StateDegrades • UpdateReceived • UpdateAccepted • UpdateRejected • UpdateFinished
IntentReport API	
Negotiate Intent	Judge Probe Best
Intent Profile	

Table 4. API notifications

8 Component API Specification

A new Intent API will be defined for this Component.

9 Administrative Appendix

9.1 References

Reference	Description	Source	Brief Use Summary
IG1253C	Intent Lifecycle Management and Interface v1.1	TM Forum	Throughout
IG1230	AN Technical Architecture v.1.1	TM Forum	Chapter 2
IG1253	Intent in Autonomous Networks v1.1.0	TM Forum	Throughout

9.2 Document History

9.2.1 Version History

Version	Date Modified	Modified by:	Description of changes
0.0.1	02-OCT-2021	Kevin McDonnell	Initial Draft
0.0.2	03-NOV-2021	Fernando Camacho	Detailed Functions
1.0.0	30-NOV-2021	Alan Pope	Formatted for publication
1.0.1	20-DEC-2021	Kevin McDonnell	Update Intent Domain Model figure and address Team review comments
1.0.2	23-DEC-2021	Kevin McDonnell	Addressed comments by Yuval Stein
1.0.3	17-JAN-2021	Kevin McDonnell	Incorporate comments by Joerg Niemoeller. Intent always manipulation in the 'aggregate' – no API operations performed on *Expectation* or *Context* objects that are inside Intent object
1.0.4	28-JAN-2021	Kevin McDonnell	AN Team Approved
1.0.5	17-JAN-2021	Kevin McDonnell	Domain Entities and Domain Events
1.0.6	28-MAR-2021	Kevin McDonnell	Address comments on Error sections from Pierre Gauthier

9.2.2 Release History

Release Number	Date Modified	Modified by:	Description of changes
Pre-production	31-Mar-2022		Ready for publication

9.3 Contributors to this Document

This document was prepared by members of the TM Forum Autonomous Network Project and the Open API project team:

- Kevin McDonnell, Huawei
- Fernando Camacho, Huawei

Additional input was provided by the following people:

- Joerg Niemoeller, Ericsson
- Kamal Mougsoudlou, Ericsson
- Azahar Machwe, BT
- Lei Wang, Huawei
- Laimin Wang, Huawei
- Yuval Stein, TEOCO
- Dave Milham, TM Forum
- Pierre Gauthier, TM Forum
- Alan Pope, TM Forum

Annex A: PlantUML source code

Intent management

Create Intent

```
@startuml
title "[Create an intent]"
actor "Intent Management \n<<Owner>>" as O
participant "Intent Management \n<<Handler>>" as H
participant "CL Management" as CL
Collections "Target" as T
O -> H: Request to create an intent instance
H -> H: Create intent object
H -> O: Response on intent creation \n (success/failure)
alt status is OperationSucceeded
H -> H: Translate intent expectations  \n into service or network actions
H -> CL: Required Actions (Config)
activate CL
loop for all requirements
   CL -> T: Evaluate requirement
   CL -> T: Adjust to fulfil the requirement
   T -> CL: ok/nok
end
deactivate CL
alt status is Intent Accepted
H -> H: Set intent status \n (compliant/degraded)
H -> O:Notify status change
else status is Intent Rejected
H -> H: Set intent status \n (finalising)
H -> O:Notify status change
loop for all requirements
   CL -> T: Rollback all changes
end
H -> H: Delete intent object
H -> O: Notify intent rejection \n Provide reasons and options

end
end
hide footbox
@enduml
```

Modify Intent

```
@startuml
title "[Modify an intent]"
actor "Intent Management \n<<Owner>>" as O
participant "Intent Management \n<<Handler>>" as H
participant "CL Management" as CL
Collections "Target" as T
O -> H: Request to modify an intent instance
H -> H: Set intent status to \nupdate received
H -> O: Response on intent update \n(sucess/failure)
alt status is OperationSucceeded
H -> H: Translate new/updated intent expectations  \ninto service or network actions
H -> CL: Required Actions (Config)
activate CL
loop for all requirements
   CL -> T: Evaluate requirement
   CL -> T: Adjust to fulfil the requirement
   T -> CL: ok/nok
end
deactivate CL
alt status is Intent Accepted
H -> H: Set intent status \n (compliant/degraded)
  H -> O:Notify status change
else status is Intent Rejected
H -> H: Set intent update status \n (No update)
H -> O:Notify status change
loop for all requirements
```

```
   CL -> T: Rollback all changes
end
H -> O: Notify intent rejection \n Provide reasons and options

end
end
hide footbox
@enduml
```

Remove Intent

```
@startuml
title "[Remove an intent]"
actor "Intent Management \n<<Owner>>" as O
participant "Intent Management \n<<Handler>>" as H
participant "CL Management" as CL
Collections "Target" as T
O -> H: Request to delete an intent instance
H -> H: Set intent status to finalising
H -> O: Response on intent removal \n(sucess/failure)
alt status is OperationSucceeded
H -> CL: Remove Actions (Config)
activate CL
loop for for all requirements
   CL -> T: Remove target entity if needed
end
deactivate CL
H -> H: Delete intent object
H -> O: Notify intent removal
end
hide footbox
@enduml
```

Judge / Preference

```
@startuml
title "[Judge / Preference Interaction]"
actor "<<Intent Owner>>" as O
box "Intent Management Function" #transparent
participant "Message Queue" as M
participant "<<Intent Handler>>" as H
end box
participant "CL Management" as CL
Collections "Target" as T
H -> M: Judge
M -> O: Notification
O -> H: Preferences
H -> CL: Required Actions \naccording to preferences
CL -> T: Adjust
hide footbox
@enduml
```

Probe Intent

```
@startuml
title "[Probe Interaction]"
actor "<<Intent Owner>>" as O
box "Intent Management Function" #transparent
participant "Message Queue" as M
participant "<<Intent Handler>>" as H
end box
participant "CL Management" as CL
Collections "Target" as T
O -> H: Probe
H -> M: Probe received
M -> O: Notification
H -> H: Translate intent expectations  \n into service or network actions
H -> CL: Enquire about requirements
activate CL
```

```
loop for all requirements
   CL -> T: Evaluate requirement
   T -> CL: ok/nok
end
CL -> H: Provide feedback
deactivate CL
H -> M: Provide Intent Report\n for Probe
M -> O: Notification
hide footbox
@enduml
```

Best / Propose

```
@startuml
title "[Best Interaction]"
actor "<<Intent Owner>>" as O
box "Intent Management Function" #transparent
participant "Message Queue" as M
participant "<<Intent Handler>>" as H
end box
participant "CL Management" as CL
Collections "Target" as T
O -> H: Best
H -> M: Best received
M -> O: Notification
H -> H: Translate intent expectations  \n into service or network actions
H -> CL: Enquire about requirements
activate CL
loop for all requirements
   CL -> T: Evaluate requirement
   T -> CL: ok/nok
end
CL -> H: Provide feedback
deactivate CL
H -> M: Propose Intent Report\n for Best
M -> O: Notification
hide footbox
@enduml
```