Intent API architecture

# A diagram of a computer Description automatically generated A diagram of a computer network Description automatically generated

From Intel/Thijs: Send to us 27/11 revised 17/12 (stored in repo/Misc).

Things to discuss:

* In the INTEND project architecture, should the Intent Manager(s) (IM) be the INTEND Tools or should they manifest themselves to be separate entities (as in your figure)?
* According to TM Forum, it is possible with a hierarchy of Intent Managers (and there will most likely be such a hierarchy (business, service, resource => refinement of intents)
* Further, if we are to follow the TM Forum way of looking at this, the communication between Intent Owners (often an Intent Manager further “up” in the hierarchy) and an Intent Manager is always an Intent (following the Intent ontology, and the Intent Common Model (ICM) of the ontology is mandatory). The ICM contains one Target and at least one Expectation (DeliveryExpectation, PropertyExpectation, ReportingExpectation). The target could be set to a tool in INTEND.
* The figure below should probably include reporting going back (up) to the intent owner. This is needed to handle lifecycle of intents.
* Which tool is handling the lifecycle of intents, or is it federated/distributed between tools?

We suggested this alternative, aligned with TM Forum:

A diagram of a company

Description automatically generated

It represents a very simplified version (open-loop) of lifecycle management and is probably not sufficient for all AI4Data use-cases since it implies manual steps if changes are needed. In truly fire and forget scenarios, this might however be sufficient for those use-cases (omitting the manual steps).

* Specify the intent as a static, predefined goal.
* Ensure the intent aligns with system capabilities and operational constraints.
* Configure the system to execute the intent as a static rule or schedule.
* **Manually** assess if the intent remains relevant and effective over time.
* **Manually** update the intent if requirements change or retire it if no longer needed.

Note that in this open-loop lifecycle management scenario “ReportingExpectations” are in many cases not present in the original intent since they, at least for lifecycle management purposes are not relevant due to the fire and forget semantic.

A more sophisticated scenario like this figure illustrates is probably more likely to be fitting for UC3:

A diagram of a process flow

Description automatically generated

In this scenario the lifecycle management could be envisioned to be something like this:

* Define and capture the high-level goal or outcome desired by the user.
* Translate the high-level intent into actionable sub-intents or policies that can be understood by the underlying systems.
* Negotiate if (chain of) intents can be accepted
* **Implement the translated intent.**
* **Continuously monitor the system to ensure the intent is being met.**
* **Continuously refine and optimize the intent's implementation to adapt to changing conditions.**
* **Enable dynamic updates to the intent as business needs or external conditions change.**
* **Conclude or deactivate the intent when it is no longer needed.**

In this scenario, “ReportingExpectations” are needed since closed-loop management of the intent lifecycle is performed.

# Implementation of intent lifecycle

Should lifecycle management be viewed as a separate piece of functionality, or should it be federated between the INTEND tools? The figure below illustrates a scenario where it is regarded as a separate functionality.

# A diagram of a diagram of a diagram Description automatically generated

As can be seen from the figure above, persistency (storage) of intents and intents reports is needed. In this figure it is illustrated as knowledge graphs in a knowledge graph database (i.e. Neo4Jj or GraphDB), since this is what TM Forum is promoting. Regardless of decisions related to: if lifecycle management is to be regarded as a separate functionality or an intrinsic part of all INTEND tools, the repository could be regarded as a central repository storing state for intent and intent reports related to the intent lifecycle management.

## First step, a Minimal Viable Product for intent lifecycle management

Let us start experimenting with a knowledge graph implementation of intent and intent reporting state. How should intents be represented in a knowledge graph? How should intent reports be modelled? How should lifecycle events be modelled? How should a closed-loop management strategy be implemented on top of the knowledge graphs?

### Step 1 Define a few simple intents

Create a few simple intents expressed following the Intent Common Model ontology defined by TM Forum. As a start, pick one of them and store it as triples in a Knowledge Graph (decide on which knowledge graph to use: Neo4j, GraphDB, ….). The selected intent should have at least one ReportingExpectation for metrics that can be used to see if the intent is being met by the underlying system. How detailed should the DeliveryExpectation be, should it be possible to directly infer when the condition is not met, or should it be up to the “logic” to decide (see Step 3)?

### Step 2 Define a way to simulate and represent intent reports

Find out how to represent the intent reporting as triples in a knowledge graph. Simulate the sending/reception of ReportingExpectation events.

### Step 3 Define a closed-loop algorithm

Create a simple closed-loop algorithm that at defined intervals checks to see if the intent is being met. If it is, nothing needs to be done. If it is not, decide what to do about it… The checking is based on intent reports that is forwarded from the underlying system (as per the ReportingExpectation). In its simplest form this could be something like this for network QoS intents:

* Make a query to retrieve all reports for all metrics that the intent should be measured over (e.g. get all latency reports)
* Check if the intent metric is met (e.g. Does all reports indicate that latency is met?)

It could also be possible to use an LLM to decide if the metric is met or not, for example instructing it (prompt): “If there are periods for more than one minute with consecutive violations of the metric: latency < 10ms, then flag it”.

### Step 4 Redefine the intent

Find out how to redefine the intent when it is not met (negotiation?). What is involved here?

### Step 5 What did we learn?

Will the intent manager have to have separate threads to handle the closed-loop lifecycle management for all intents?

Do we need to have cross-cutting actions to see if there are conflicting events?

Do we need to model the capabilities of the underlying system (metrics), i.e. create a digital twin?

# Property graph database or triple store?

To store and manage the **TM Forum Intent Ontology** (expressed in **Turtle files**) and enable **closed-loop lifecycle management of intents**, a **triple store database** is the most suitable choice. Turtle files use the **RDF** (Resource Description Framework) standard to represent linked data, which triple stores are designed to handle natively. Triple stores are optimized for storing and reasoning over such semantic data. They support reasoning and inference, enabling the dynamic deduction of relationships or classification of intents based on predefined rules. Triple stores use **SPARQL**, a W3C-standardized query language, to query RDF data. This is ideal for querying and manipulating the intents and intent reports in a structured and semantic manner. RDF data stored in a triple store adheres to open standards, ensuring interoperability with other tools, APIs, or systems within the TM Forum framework or beyond. A triple store can:

* Manage and query **intents** and **intent reports** as triples.
* Leverage reasoning capabilities to identify discrepancies, automate actions, or infer updates to intents based on intent reports.
* Maintain a historical view of intent changes and their associated lifecycle states, essential for closed-loop operations.
* Integrate the triple store with automation frameworks (e.g., workflow engines, AI systems) to trigger **closed-loop actions**. For example, an intent report might automatically update an intent's lifecycle state and trigger a workflow for resolution.
* Use SPARQL queries to monitor intent statuses, generate reports, and identify patterns or trends.

A **triple store database** is the best choice for implementing a system based on the **TM Forum Intent Ontology** and enabling **closed-loop lifecycle management of intents**. Tools like **GraphDB** or **Apache Jena** provide the native RDF and SPARQL support, reasoning capabilities, and interoperability needed for such a semantically rich use case.

# SHACL Shapes for the 5G4DATA use-case

We have decided to use the relevant parts of the TMForum intent ontology to express intents for the 5G4DATA use-case. How should intents for the use-case be formed? Even though we have an ontology as a baseline, it might be that our use-case does not need all of the Classes or Properties defined in the ontology, at leas not in a first version PoC/MVP. It is probably a good idea to (continuously) develop SHACL shapes for legal intents for the 5G4DATA uses-case. Note that SHACL shapes can vary for different use cases. SHACL (Shapes Constraint Language) is highly flexible and allows for tailored validation rules that can align with the needs of a particular context or application. This modularity and adaptability make SHACL a powerful tool for enforcing constraints and validating RDF data.

## How to Design SHACL Shapes for the 5G4DATA use-case

SHACL shapes can be tailored for different use cases to validate the same ontology in varied ways. This flexibility ensures that the same RDF dataset can be validated according to different requirements without modifying the ontology itself. This is particularly useful in dynamic, multi-application environments where diverse validation rules are necessary.

First of all, we need to Identify Use Case Requirements related to the entities, relationships, and constraints from the TM Forum ontology that are critical for the use-case, including which constraints are mandatory and which are optional. Cardinality of predicates is also something that needs to be considered.

# Lifecycle management tasks

These are some of the aspects that should be considered when creating an intent manager.

An **Intent Manager** in the context of **TM Forum intents** plays a critical role in managing and fulfilling intent-driven operations, particularly in intent-based networking (IBN) and business processes. Below are the core responsibilities and tasks that an Intent Manager should **Capture and Understand Intents** and Translate these intents into actionable objectives that align with the organization's operational goals. Received intents should be checked for syntactical and semantical validity. An intent manager should Identify and resolve conflicts between multiple intents or between an intent and existing policies. The intent manager should Decompose high-level intents into sub-intents or specific technical tasks and translate intents into workflows or commands understandable by underlying systems, such as orchestrators, controllers, or AI models. In order to manage the life cycle of intents it is necessary to continuously monitor system states and metrics to ensure intents are being fulfilled and initiate corrective actions if deviations are detected. If (or when) deviations are detected, the intent manager should dynamically adapt intents based on the changing operational conditions or evolving user requirements. Part of the work of an intent manager includes to offer transparent updates to users about the status of intent fulfillment (explainability). In short, an intent manager needs to handle the complete lifecycle of an intent, from submission to fulfillment and termination. The **Intent Manager** acts as a bridge between high-level goals (intents) and the underlying systems that execute them. It ensures alignment between business objectives and technical operations by automating intent interpretation, decomposition, orchestration, and assurance while continuously learning and adapting to new challenges. This role is crucial in advancing toward autonomous networks and operations in accordance with TM Forum standards.

# Identified problems/research questions

## Formal specification for intent

We are trying to create something that is intent driven (do actions based on expressed intents). An intent states what you want to achieve and not how to do it. In order for the handling of intents to be done automated it is necessary to define a formal definition for intents that is easily machine readable and possible to reason over for machines. What should this formal definition look like? Should we create our own, or should we use something that already exists (e.g. TM Forum Intent ontology).

## Accuracy of generative AI

We want to transform intents expressed in natural language to formal intent descriptions (using whatever formal specification we decide on). In general, this is a very difficult task, but for limited scenarios (like the Intend project use-cases) this might be possible? We have investigated to use generative AI (e.g. LLM) to do this translation, but we do observe that the generative AI models are a bit to “imaginative” and creating output that is far from perfect (100% accuracy). How do we handle this? Some directions coultd be RAG, RAT, fine tuning, Mixture of Experts, guardrails (e.g. use SHACL shapes to verify syntax), ….