***Disclaimer***

The present document has been produced and approved by the Experiential Networked Intelligence (ENI) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.  
It does not necessarily represent the views of the entire ETSI membership.

ETSI GR ENI 015 V0.0.18 (2023-12)

**Group REPORT**

Experiential Networked Intelligence (ENI);

Processing and Management of Intent Policy

<

Reference

DGR/ENI-0025

Keywords

Artificial Intelligence, Network

***ETSI***

650 Route des Lucioles

F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B

Association à but non lucratif enregistrée à la

Sous-préfecture de Grasse (06) N° w061004871

***Important notice***

The present document can be downloaded from:  
<http://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at [www.etsi.org/deliver](http://www.etsi.org/deliver).

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at <https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>

If you find errors in the present document, please send your comment to one of the following services:  
<https://portal.etsi.org/People/CommiteeSupportStaff.aspx>

If you find a security vulnerability in the present document, please report it through our

Coordinated Vulnerability Disclosure Program:

<https://www.etsi.org/standards/coordinated-vulnerability-disclosure>

***Notice of disclaimer & limitation of liability***

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or

other professional standard and applicable regulations.

No recommendation as to products and services or vendors is made or should be implied.

No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.

In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

***Copyright Notification***

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.  
The content of the PDF version shall not be modified without the written authorization of ETSI.  
The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2023.

All rights reserved.

Contents

Intellectual Property Rights 4

Foreword 4

Modal verbs terminology 4

1 Scope 5

2 References 5

2.1 Normative references 5

2.2 Informative references 5

3 Definition of terms, symbols and abbreviations 6

3.1 Terms 6

3.2 Symbols 6

3.3 Abbreviations 6

4 Background and Overview 6

4.1 ENI Purpose 6

4.2 Intent Work in ENI 7

4.3 Introduction to Knowledge 7

5 Procedures of Intent Policy Processing 7

5.1 Introduction 7

5.2 Intent Policy Processing Operation 7

5.3 Conflict detection and resolution 9

5.3.1 Overview 9

5.3.2 Conflict detection 9

5.3.3 Conflict resolution 11

6 Knowledge management for Intent Policy 12

6.1 Introduction to Knowledge Graphs 12

6.1.1 Definition 12

6.1.2 Motivation for Using Knowledge Graphs in ENI 12

6.1.2.3 Location of the Knowledge Graph 13

6.2 Constructing Knowledge Graphs in the ENI System 14

6.2.1 Introduction 14

6.2.2 Construction Procedures 14

6.2.2.1 Data Processing 14

6.2.2.2 Knowledge Representation 15

6.2.2.3 Knowledge Generation 15

6.2.2.4 Knowledge Verification 16

6.2.2.5 Knowledge Graph Storage 16

6.3 Using Knowledge Graphs to Manage Intent policies 16

6.3.1 Intent Translation 16

6.3.2 Intent Verification 17

6.3.3 Intent Monitoring 17

6.3.4 Intent Assurance 17

6.3.5 Intent Optimization 17

6.4 Lifecycle Management 17

6.4.1 Lifecycle Management of Intent Policies 17

6.4.1.1 States of Intent Policy management 17

6.4.1.2 State machine of Intent Policy processing 18

6.4.1.3 Lifecycle Management Roles of Intent Policy 19

6.4.2 Lifecycle Management of Knowledge Used by Policies 19

6.4.2.1 Previous Work 19

6.4.2.2 Lifecycle Management Roles of Knowledge Used by Policies 20

6.4.2.3 Lifecycle of Knowledge Used by Policies 20

7 Use Cases 21

7.1 Use cases for operators’ business 21

7.2 Use cases for vertical industry 23

8 Conclusions 26

Annex : Change History 27

History 28

# Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server ([https://ipr.etsi.org](https://ipr.etsi.org/)).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

**DECT™**, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

# Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group <long ISGname> (<short ISGname>).

# Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the [ETSI Drafting Rules](https://portal.etsi.org/Services/editHelp!/Howtostart/ETSIDraftingRules.aspx) (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

# 1 Scope

The present document describes the following topics:

* Enhanced procedures for processing Intent Policy, e.g.:
* Detail the Procedures of intent policy processing;
* Conflict detection and resolution between different Intent Policies;
* Knowledge management for Intent Policy, including:
* How to use a Knowledge Graph to manage Intent policies;
* How to use a Knowledge Graph for managing Intent policy knowledge;
* Procedures for lifecycle management of intent knowledge, e.g. import, update, delete, and query of the intent knowledge.
* Typical use cases and requirements which can reduce the management complexity for Intent Users, e.g.:
* Use cases for Business Users/Operational Users/Technical Users that are all users of Intent.

# 2 References

## 2.1 Normative references

Normative references are not applicable in the present document.

## 2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non‑specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GS ENI 005 (V3.1.1): "Experiential Networked Intelligence (ENI); System Architecture".

[i.2] ETSI GR ENI 008 (V2.1.1): "Experiential Networked Intelligence (ENI); InTent Aware Network Autonomicity (ITANA)".

[i.3] ETSI GR ENI 016 (V2.1.1): "Experiential Networked Intelligence (ENI); Functional Concepts for Modular System Operation".

[i.4] ETSI GR ENI 030 (V3.1.1): "Experiential Networked Intelligence (ENI); Transformer Architecture for Policy Translation".

[i.5] Chen, Zhe, et al. "Knowledge graph completion: A review" IEEE Access 8 (2020): 192435-192456.

[i.6] TM Forum IG1253 (V1.2.0): "Intent in Autonomous Networks".

[i.7] MEF 95: "MEF Policy Driven Orchestration", Strassner, J, editor, July 2021. NOTE: Available at: <https://www.mef.net/resources/mef-95-mef-policy-driven-orchestration>.

[i.8] ETSI GS ENI 019 (V3.1.1)”: "Experiential Networked Intelligence (ENI); System Architecture: Representing, Inferring, and Proving Knowledge in ENI".

[i.9] ETSI GS ENI 033 (V4.1.1)”: "Experiential Networked Intelligence (ENI); Definition, Requirements and Procedure of Intent Policy Multi-Stage Translating".

# 3 Definition of terms, symbols and abbreviations

## 3.1 Terms

For the purposes of the present document, the terms apply:

## 3.2 Symbols

For the purposes of the present document, the [following] symbols [given in ... and the following] apply:

## 3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply::

|  |  |
| --- | --- |
| ACL | Access Control List |
| API | Application Programming Interface |
| BSS | Business Support System |
| CPL | Cloud Private Line |
| CSRUD | Create-Store-Read-Update-Delete |
| GDB | Graph Database |
| IDS | Intrusion Detection Systems |
| IPFB | Intent Parser Functional Block |
| KG | Knowledge Graph |
| LSA | Latent Semantic Analysis |
| NAS | Network Attached Storage |
| NER | Named Entity Recognition |
| NLP | Natural Language Processing |
| O&M | Operation and Maintenance |
| OPEX | Operational Expenditure |
| OSS | Operations Support System |
| OTN | Optical Transport Network |
| PMFB | Policy Management Functional Block |
| QoS | Quality of Service |
| RAN | Radio Access Network |
| RDF | Resource Description Framework |
| UHD | Ultra High Definition |
| WOL | Web Ontology Language |
| W3C | World Wide Web Consortium |
| XR | Extended Reality |

# 4 Background and Overview

## 4.1 ENI Purpose

Operational Expenditure (OPEX) for network management is one of the operators’ biggest concerns. The use of intent in ENI can help operators effectively reduce OPEX. Intent enables different constituencies to express intent policies using a language that is natural to themselves. Hence, the associated business benefit is understanding business needs and using them to determine offered services and resources. This means that offered services can dynamically adapt to changing user needs, business goals, and environment conditions, which will greatly improves the efficiency of network management and reduces the cost of network O&M (Operation and Maintenance) .

## 4.2 Intent Work in ENI

Intent Policy expresses the goals to be accomplished by using a restricted natural language (e.g., an external DSL), without focusing on how to achieve those goals. The definition and introduction about intent policy can be found in clause 6.3.9.3 of ETSI GS ENI 005 [i.1], and a previous study (ETSI GR ENI 008 [i.2]) further discussed the architecture of intent policy（in clause 5.1.2）, life cycle management (in clause 5.3) and the translation process of intent policy (in clause 5.2.3), as well as related use cases (in clause 6).

## 4.3 Introduction to Knowledge

Knowledge is defined in clause 3.1 and further detailed in clause 6.3.4 of ETSI GS ENI 005 [i.1]. Briefly, knowledge analyzes data and information, understands its meaning, and can predict what has happened, is happening, or is possible to happen in the future. Knowledge management is an essential part in intent aware network, which is responsible for managing the life cycle of the knowledge in the Data and Knowledge Repositories, including storage, assessment, use, sharing, and refinement of knowledge assets. ENI develops knowledge and wisdom from observed, measured, and inferred data and information, as described in [i.3].

# 5 Procedures of Intent Policy Processing

## 5.1 Introduction

In GR ENI 008 [i.2] and GS ENI 005 [i.1], the procedures of intent policy translation are shown in different detail. In general, the intent policy is created by the intent creator and sent to the ENI system. The ENI system translates and processes the received intent policy and notifies the intent creator of any errors. If there are no errors, then the intent policy is executed on the selected component(s) of the Assisted System.

Considering the complexity of user’s goals, the intent policy can be separated into multiple intent policies to simplify the general procedures of intent policy processing, such as translation and named entity recognition. The processed intent policy has three main lifecycle phases (see clause 6.5 of the present document): Translation, Deployment, and Execution.

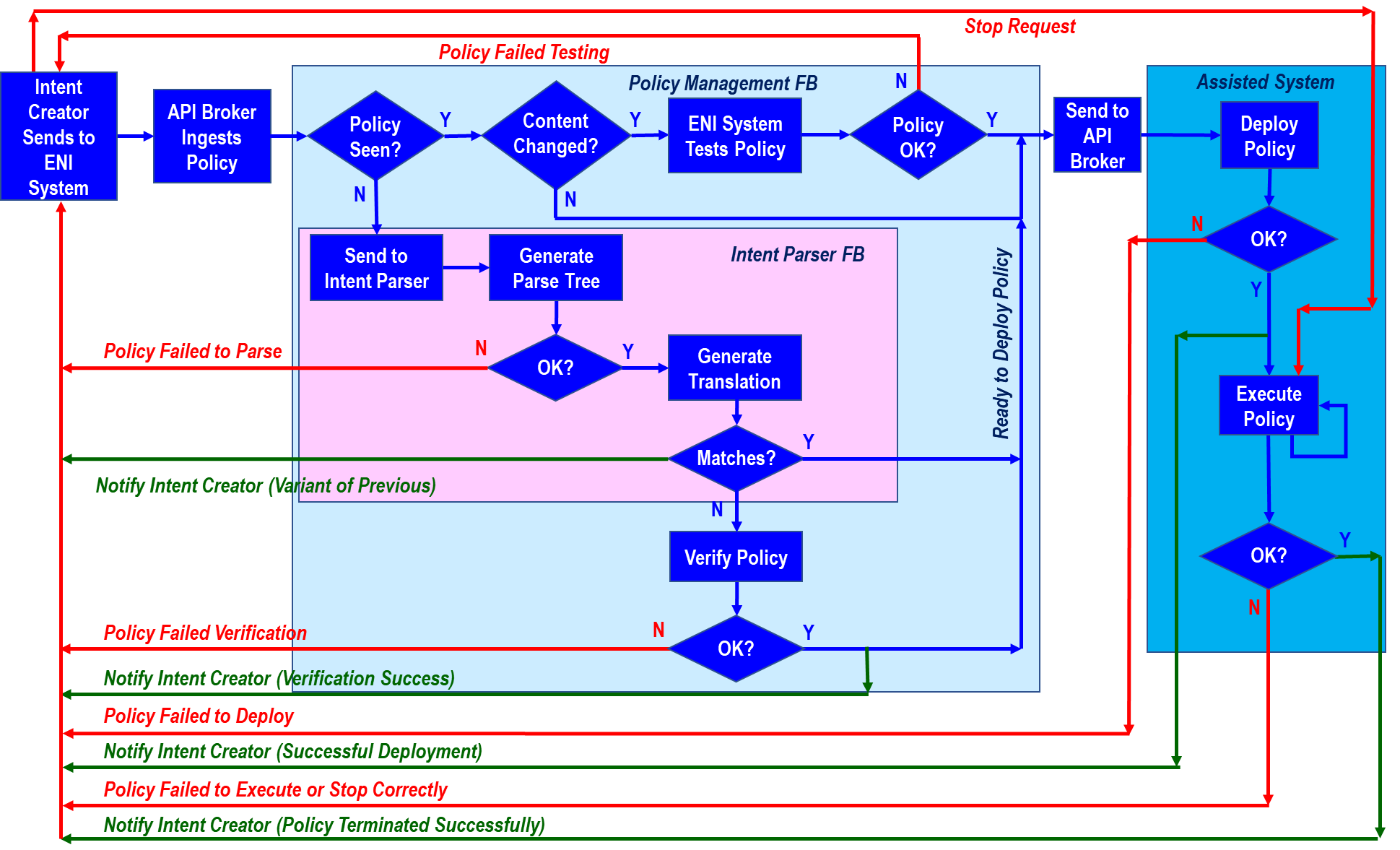
The benefits of this process are as follows:

1. Ensure the realization of the intent creator’s goals by performing two functions automatically:
2. Translating the intent creator’s policy from natural language to a form that can be verified by the ENI System. Subsequent translations could be performed to make the translated policy implementable by the Assisted System. If the translation fails, then errors found will be sent to the intent creator, and processing stops until the intent creator resubmits an edited intent policy.
3. Testing any new intent policies to ensure that they can be executed correctly. If the testing results are not able to meet the goals defined by the intent creator, then information describing this will be sent to the intent creator, and processing stops until the intent creator resubmits an edited intent policy.
4. Automate the entire process (all processes are triggered automatically, especially automatic testing), while still providing flexibility to the intent creator. For example, the intent creator can have control on each state change of the intent policy lifecycle (see clause 6.5).
5. Support the flexible use of intent policies that are managed by a knowledge process, such as the Knowledge Management Functional Block of GS ENI 005 (see clause 6.3.4 of [i.1]).

NOTE: a group of intent policies could also be managed as a unit (e.g., a high level intent with an associated set of detailed policies on each resource).

## 5.2 Intent Policy Processing Operation

The original Intent Policy, which is written in a restricted natural language, is sent to the ENI System. It is ingested by the API Broker and sent to the Policy Management Functional Block (PMFB). The PMFB checks to see if the ingested Intent Policy has been seen before (e.g., by matching the text of the newly ingested Intent Policy with the text of a previously stored Intent Policy). If the original Intent Policy matches a previous Intent Policy that has been successfully parsed, verified, and stored, then the ENI System checks to see if anything has changed in the ingested Intent Policy (e.g., parameter values). If nothing has changed, then the PMFB retrieves the previously translated representation of the matched Intent Policy and sends it to the API Broker. The API Broker transforms the translated Intent Policy to a form (i.e., a set of objects and/or code) that the Assisted System can understand. The translated Intent Policy is then sent to the Assisted System. If a change in the ingested Intent Policy is detected, then the PMFB will first verify the proper operation of the Intent Policy by testing it before sending it to the Assisted System. If this test fails, then all errors and warnings are collected and sent back to the Intent Creator. If it passes, then the PMFB sends the verified Intent Policy to the API Broker as described above.



**Figure 5-1 Detailed Procedures of Intent Policy Processing**

Alternatively, if the original Intent Policy was not found, then it needs to be parsed, optionally compiled and translated, and subsequently verified. This is shown in the rest of Figure 5-1, starting with the “Send to Intent Parser” block. The steps shown in Figure 5-1 are summarized as follows.

1. The input Intent Policy is parsed by the Intent Parser, which is part of the Intent Parser Functional Block (IPFB). The parsing process is described in detail in in ENI 005 [i.1], translator architecture in ENI 030 [i.4] and ENI 008 [i.2].
   1. If the Intent Parser fails to generate a parse tree, then all errors and warnings are collected and sent to the Intent Creator for correction.
   2. Otherwise, the parse tree is generated and sent to the Intent Translation function, which is another Functional Block that resides in the IPFB.
2. The Intent Translation function checks to see if the Intent Policy has been previously used. Conceptually, the Translation function is matching the newly translated object form of the ingested Intent Policy with the object form of the stored Intent Policy, The purpose of this check is to catch grammatical differences that resolve to the same Policy. For example, if Intent Policy 1 says “Give Ray Gold Service” and Intent Policy 2 says “Give Gold Service to the user whose IP Address is 10.10.10.1”, and the user of that IP address is Ray, then Intent Policy 1 and Intent Policy 2 are the same. The translation process is specified in ENI 005 [i.1], translator architecture in ENI 030 [i.4] and described as intent translation in ENI 008 [i.2].
   1. If it has, then the Intent Translation function notifies the Intent Creator and the Intent Policy is ready to be deployed by the Assisted System (step 4).
   2. If not, then the Intent Translation function passes the translated Intent Policy to the PMFB to verify the Intent Policy (step 3).
3. The Intent Policy is verified by testing it (either by simulation or in a closed sandbox). In either case, testing consists of a set of test inputs, execution conditions and expected results prepared for a specific objective, which is used to verify whether the goals of the Intent Policy are met.
   1. If the Intent Policy fails verification, then all errors and warnings are collected and sent to the Intent Creator for correction.
   2. Otherwise, the Intent Policy is ready to be deployed by the Assisted System.
4. The Intent Policy is deployed when either the Intent Creator or the ENI System give the Intent Policy Processing system permission to do so. (The ENI System allows either the Intent Creator to have direct control over the deployment of the Intent Policy, or alternatively, to tell the ENI System to automatically deploy it when ready; this latter is a configurable setting).
   1. If the Intent Policy fails to be deployed, then all errors and warnings are collected and sent to the Intent Creator for correction.
   2. Otherwise, the Intent Policy is ready to be executed.
5. The Intent Policy is executed when either the Intent Creator or the ENI System give the Intent Policy Processing system permission to do so. (The ENI System allows either the Intent Creator to have direct control over the execution of the intent policy, or alternatively, to tell the ENI System to automatically execute it when ready; this latter is a configurable setting).
   1. If the Intent Policy fails to execute, then all errors and warnings are collected and sent to the Intent Creator for correction.
   2. Otherwise, the Intent Policy executes.

The Intent Policy will continue to execute until either the Intent Creator or the ENI System stop its execution.

The above process does not include policy conflict detection and remediation. This is detailed in Clause 5.3.

## 5.3 Conflict detection and resolution

### 5.3.1 Overview

Conflicts of policies are a well-known challenge to any policy management. The ENI system usually receives intent policies from multiple sources (OSS, BSS, application, end-user, orchestrator, etc.). The ENI System will detect whether there is a conflict between the new Intent Policy and other Intent Policies in Policy Management Functional Block. If the conflict is detected, then the ENI system and Intent Creator try to resolve the conflict. Conflict detection and resolution between different Intent Policies will be discussed in this section.

NOTE: This document will only discuss conflict detection and resolution between intent policies. Conflict detection and resolution between intent policies and other types of policies will be done in a future version of the present document.

### 5.3.2 Conflict detection

When a new intent policy comes, the ENI system needs to consider the possible conflict between it and any existing intent policies that are currently running. If any conflict is detected, the conflict message will be sent to the intent creator, and the knowledge management function block records the conflict information for further reasoning and analysis. The policy conflict is defined in ETSI GS ENI 005 [i.1] as follows: Policy conflict means two policies that, when executed, cause contradictory and otherwise incompatible results within a given execution time window.

From the perspective of detection, there are two levels of conflicts (refer to GS ENI 005 [i.1] and TM Forum IG1253[i.6]):

* **Direct conflict**

Direct conflict means that Intent Policies are expressing explicitly opposing or incompatible requirement with each other. The most important thing in this case is to confirm the execution domains and time.

The first example is the following:

* Example 1

***Intent Policy A: All network links are encrypted.***

***Intent Policy B: All network links are not encrypted.***

As written, and with no other information, Intent Policy A and Intent Policy B conflict. However, for example, as long as the user groups do not overlap, there is no explicit contradiction. This can for example be a non-overlapping scope such as requiring encrypted links for one user group and no encryption for another user group, so they can coexist.

Another example:

* Example 2

***Intent Policy C: Set the value of an attribute named numErrors to “2” during 08:00:00 to 08:30:00.***

***Intent Policy D: Set the value of an attribute named numErrors to “3” during 08:45:00 to 09:00:00.***

In this case, Intent Policy C and Intent Policy D do not conflict because their execution times do not overlap, while they will conflict when the execution time overlaps.

* **Indirect conflict**

In many cases, conflicts can not be obviously seen from the Intent Policy itself, but originate from conflicting actions being proposed in their handling process, which is generally caused by common infrastructure and limited resources.

* Example 3:

***Intent Policy E: Increase RAN coverage delivered to users.***

***Intent Policy F: Increase throughput delivered to users.***

On the surface, they do not involve the same indicator and there seems no conflict. However, both Intent Policies imply opposing reconfiguration and actions of RAN cells. This is like a seesaw. When the fulfilment of one Intent Policy is improved, the other will be degraded.

* Example 4:

***Intent Policy G: Allocate 10M bandwidth to user A.***

***Intent Policy H: Allocate 5M bandwidth to user B.***

When there is only 10M available bandwidth in the network, the two Intent Policy cannot be satisfied at the same time, so conflicts occur.

Based on above, figure 5-2 shows the procedure of conflict detection, which includes:

1. Translate the new Intent Policy;
2. Detect whether the execution domain of new Intent Policy overlaps with other Intent Policies.
3. If the answer of Step 2 is yes, then detect whether the execution time of new Intent Policy overlaps with other Intent Policies.
4. If the answer of Step 3 is yes, then determine whether execution of the new Intent Policy conflicts with the actions of other active Intent Policies.

Note: The detection content in step 4 is not limited to configuration parameters and network resources. It can be multiple aspects that may conflict during the execution process of the Intent Policy.

1. If the answer of Step 4 is yes, the new Intent Policy is judged to conflict with the existing policy, and conflict resolution is needed.



Figure 5-2: The procedure of conflict detection

### 5.3.3 Conflict resolution

The ENI System needs to resolve conflicting Intent Policies, helping users to properly choose between Intent Policy alternatives that may have different ramifications.

In some conditions, conflict can be easily resolved (for example, the requirement of one Intent Policy includes another). In this case, we suppose one Intent Policy requires a maximum delay of 10ms and another one for the same target requires a maximum delay of 5ms. As written, the policies conflict, because they apply to all applications, and the system does not know whether to set the maximum delay to 5ms or 10ms. There are at least two different solutions to resolving this conflict. The first specifies which applications receive the 5 ms delay and which receive the 10ms delay. The second is to simply choose the 5 ms delay as long as all applications can support this more stringent requirement.

For more complicated situations, priority is an efficient way to resolve conflicts as described in “MEF 95: MEF Policy Driven Orchestration” [i.7]. This indicates that an ENI system should be able to prioritize actions and therefore find the set of operational states that is globally preferential even if it means to degrade some Intent Policies.

Factors affecting priority setting may include:

* Task type. For example, network failure, business configuration, engineering change, security events, etc.
* Impact on network. The tasks that have a great impact on the stability of the network will be processed first, and the tasks that have a small impact on the stability will be processed when they are idle.
* Task deadline. Prioritize tasks with deadlines approaching.

A proper scheduling algorithm should be adopted and used for priority setting and conflict resolution. The specific scheduling algorithm is beyond the scope of the present document.

# 6 Knowledge management for Intent Policy

## 6.1 Introduction to Knowledge Graphs

### 6.1.1 Definition

A Knowledge Graph is a directed acyclic graph that embeds semantics in the nodes and edges using a formal logic (RDF as a minimum; types of Description Logics, such as an OWL 2 Profile, are preferred).

NOTE: 1) cyclic graphs have loops, which means the knowledge cannot be resolved, and 2) knowledge graphs represent knowledge in a hierarchical manner.6.1.2 Motivation for Using Knowledge Graphs in ENI

This clause describes the motivation for using knowledge graph to manage intent policies, the benefits that it provides and the requirements it gives to ENI system.

**6.1.2.1 Motivation**

Knowledge graph technology is a typical cross-field technology. It is integrated with artificial intelligence, mathematics, graphics, database, and other disciplines. Based on graph theory and the probability graph model [i.5], it can identify, discover and infer the complex relationships between concepts and how they relate to collected data. In terms of using a knowledge graph for intent policy management, it is necessary to transform intent policies constructed using a restricted natural language into a structured knowledge representation, and then select Intent Policies in the knowledge graph to analyse for various management purposes. This process can help an ENI System understand the meaning of the ingested intent policy by using procedures such as named entity recognition and removing any ambiguities found in the translation process. This in turn can help translate abstract intent policies to more concrete intent policies that can be understood by the target(s) of the intent policy. The following provide examples of how a Knowledge Graph can be used to find intent conflicts.

1. Ingested intent policies can be examined in terms of their nodes and relationships to find conflicts. In addition, Metadata could be generated to automatically categorise ingested intent policies according to their nodes and relationships.
2. Knowledge Graphs (KGs) are a way of structuring information in a graph form by representing entities (e.g., customers, services, and places) as nodes, and relationships between those entities (e.g., a customer has a service located at his home) as edges. This inherent capability enables KGs to travel relationships much more efficiently than other technologies, such as directories or relational databases. When a new intent policy is ingested, the KG can verify whether there is a conflict relationship between the new intent policy and existing active intent policies. A KG can help locate and verify the cause of the conflict by using its formal reasoning capabilities. This information can then be supplied to other Functional Blocks in the Policy Management Functional Block to provide detailed information to the ENI System and to the intent creator for further conflict resolution. Conflict detection and resolution capabilities vary, and are dependent on the particular description logic used.
3. An intent policy can be separated into multiple constituent intent policies, and multiple intent policies can be combined into a composite intent policy to realize flexibility and promote reuse. All original intent policies, including any policies that are created by separating a complex intent policy into simpler intent policies, are stored as objects. This enables them to be reused to create new, more powerful, intent policies. The objects can take the form of triples in the Knowledge Graph and ENI Policy Rule objects as defined in GS ENI 019 [i.8].

NOTE: The decomposition method can be based on the semantics of the intent policy. For example, a single ingested intent policy could be translated into a hierarchical set of intent policies to remove ambiguity; this would also promote reusability, since every intent policy can be saved as an object. It could also be based on behaviour (e.g., action execution order or different operation domains or different target objects).

**6.1.2.2 Requirements**

The intent policy KG has the characteristics of large scale, complex knowledge structure, diverse sources, high dynamics and timeliness, and deeper reasoning mechanisms. These characteristics give rise to the following requirements for an ENI system:

1. It is expected to be able to use the ENI Information Model [i.8] to aid in constructing the KG. For example, classes and relationships defined in the ENI Information Model could correspond to named entities and edges in a KG, respectively.
2. An ENI System is expected to provide the ability of processing semi-structured and unstructured data into a structured knowledge representation suitable for expressing a KG. This will likely be done by a combination of the data ingestion and normalization Functional Blocks.
3. An ENI System is expected to provide the ability to construct, store and update KGs in a knowledge repository. This will likely be done by using the Repository Management Functional Block in the Knowledge Management Functional Block.
4. An ENI System is expected to provide intelligent search and match abilities for querying KGs. This will be done in the Intent Policy Processing and Management Functional Block, or equivalent Functional Blocks, which reside in the Policy Management Functional Block.
5. An ENI System is expected to provide knowledge reasoning and knowledge fusion. This will likely be done using a combination of the Knowledge Management and Cognition Management Functional Blocks. It is likely to be supported by other Functional Blocks as well.

### 6.1.2.3 Location of the Knowledge Graph

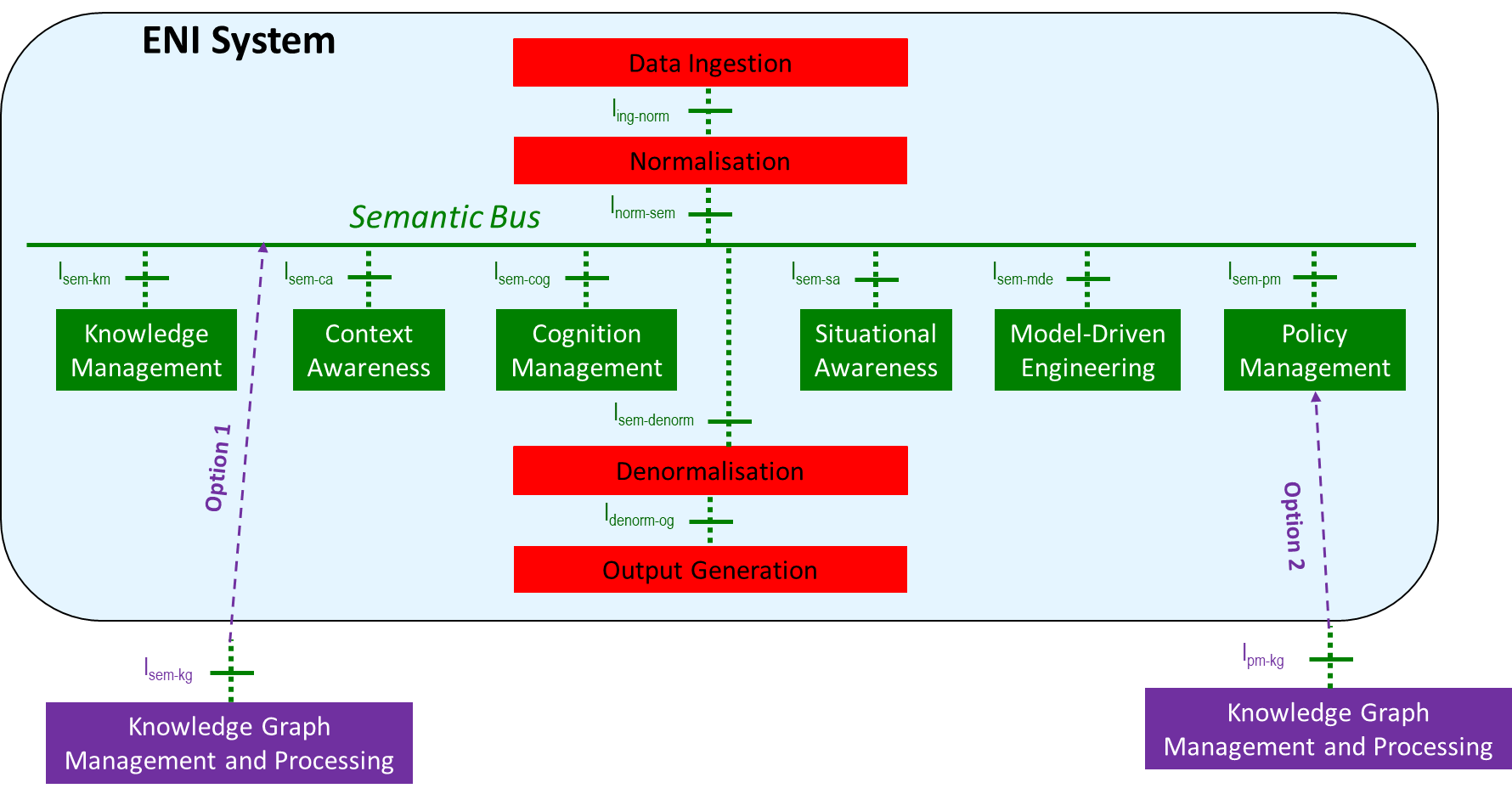
It is recommended that the Knowledge Graph processing and management be located in a Functional Block in order to be consistent with the existing ENI System architecture. Figure 6-1 shows two possible deployment options.

Figure 6- . Two Deployment Options for a Knowledge Graph Functional Block for ENI

The option on the left of Figure 6-1 creates a new top-level internal Functional Block. This option is best suited if the Knowlege Graph Functional Block will be used generically, and not specifically for ENI Policies. The option on the right of Figure 6-1 creates a new second-level internal Functional Block within the Policy Management Functional Block. This option is best suited if the Knowlege Graph Functional Block will be used specifically for ENI Policies.

Both options enable the Knowledge Graph Functional Block to take advantage of the Semantic Bus, and hence, it can communicate with the other internal Functional Blocks. Both options also prevent direct access of the Knowledge Graph from external consumers (i.e., all communication first goes through the API Broker, then the Data Ingestion and Normalization Functional Blocks, and then put onto the Semantic Bus).

## 6.2 Constructing Knowledge Graphs in the ENI System

### 6.2.1 Introduction

A Knowledge Graph is made up of three main components: nodes, edges, and labels. These components provide a mechanism to explicitly define the semantics of an Intent Policy. In addition, a Knowledge Graph has a syntax associated with it, which is the well-formedness of the underlying language (e.g., RDF or OWL) that is used to represent its entities and relations. In addition, a knowledge graph explicitly defines the semantic relationships between the various parts of an Intent Policy. Technologies related to knowledge representation and reasoning, information extraction, natural language processing, data mining and machine learning are involved in the construction of knowledge graph. One of the possible construction processes of a knowledge graph can be summarized as: process data obtained from different sources, determine the knowledge representation model, extract entities and relationships from the processed data, and then generate knowledge, verify knowledge and store knowledge for the use of ENI system.

### 6.2.2 Construction Procedures

The construction procedure of Knowledge Graphs in the ENI system includes: data processing, knowledge representation, knowledge generation, knowledge verification and knowledge graph storage. The detailed procedures are described in the following clauses, as shown in Figure 6-2

NOTE: Figure 6-2 is based on the selection of option 1 in Figure 6-1.

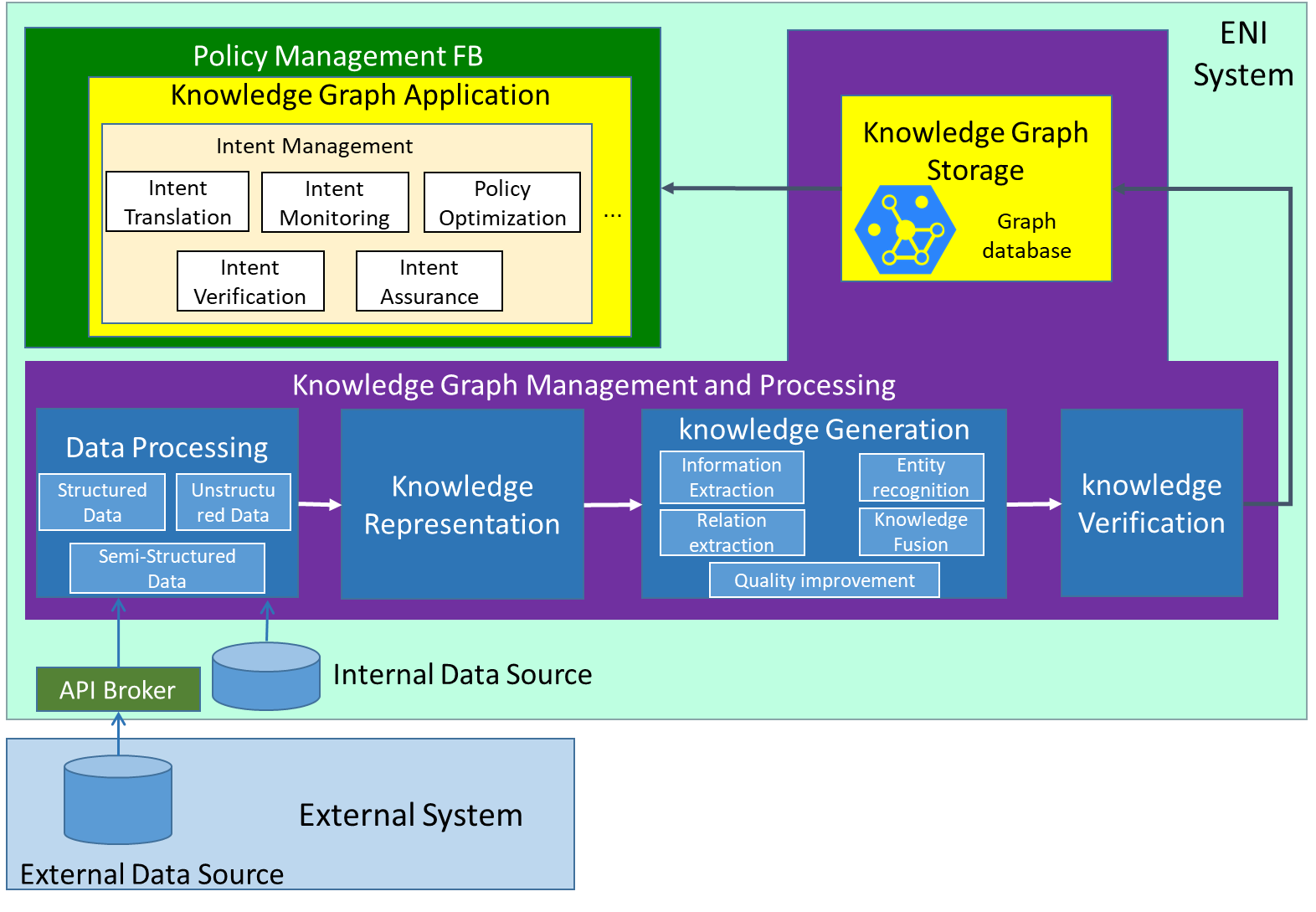


Figure 6-2: The construction procedure of Knowledge Graphs in the ENI system

#### 6.2.2.1 Data Processing

In this step, data is pre-processed (data cleaning, data filtering, etc.) and then converted to a unified format for further processing..

The data sources required for building knowledge graphs in ENI system include:

• Internal data source inside ENI system, including structured data, semi-structured data, unstructured data;

• External data source from external system, including structured data, semi-structured data, unstructured data. The data is firstly sent to ENI API Broker for format conversion, and then sent to the Data Ingestion Functional Block.

#### 6.2.2.2 Knowledge Representation

The common knowledge representation methods of knowledge graph mainly use symbolic representation methods. The symbolic representation methods include description logic, semantic network, production rules and framework system. RDF (Resource Description Framework), OWL (Web ontology language) and other similar knowledge representation methods commonly used in symbolic representation methods are recommended for the knowledge representation methods in this document. OWL has a semantic data exchange standard and specification promoted by W3C (World Wide Web Consortium). OWL has a strict semantic logic foundation, supports reasoning, and is compatible with the less complex representation language in RDF. When RDF triples (SPO) cannot meet the requirements of semantic expression, OWL, as a complete ontology language, can provide more powerful semantic expression components. These are stable and mature standards with broad application in knowledge management scenarios.

NOTE: Other knowledge representation methods (e.g., artificial neural networks) can also be used as long as they express intent as knowledge graphs in a way that is compatible with the proposed concepts in this document.

#### 6.2.2.3 Knowledge Generation

Knowledge generation includes vocabulary extraction, entity recognition, relation extraction, knowledge fusion and quality improvement.

**• Information extraction**

Information extraction refers to the process of identifying and extracting words and phrases with potential semantic information from text data. Typical methods used in information extraction include:

1. **Bag-of-words model**: represents text data as a collection of words, regardless of the order of words. The bag-of-words model can determine the important words in the text by counting the number of occurrences of each word in the text data.
2. **Word Embedding**: Representing words as low dimensional vectors that can be used for tasks such as text classification and sentiment analysis. Word embedding can be obtained through neural network learning or manually compiled.
3. **Topic model**: Identify topics or keywords in text data, such as Latent Semantic Analysis (LSA) based on probabilistic topic model.
4. **Part of Speech Tagging**: Determine the part of speech of words to better understand the text.

**• Entity recognition**

Named Entity Recognition (NER) refers to the identification of entities with specific meanings in data such as text or images, such as person names, organizational structures, place names, etc.

**• Relation extraction**

Relationship extraction is the process of extracting relationships between entities from raw data. This includes character pattern based extraction methods, grammar pattern based extraction methods, and semantic based extraction methods.

**• Knowledge fusion**

Knowledge fusion refers to the integration of knowledge from different sources, forms, and structures. By using conflict detection, consistency checking, knowledge reasoning, and other methods to determine the correctness of knowledge. The verified knowledge is aligned and linked to entities to form a more complete, accurate, and reliable knowledge base.

**• Quality improvement**

The quality improvement of knowledge graph refers to improving the accuracy, timeliness, completeness, reliability, and robustness of knowledge graph through a series of technical means and algorithms.

#### 6.2.2.4 Knowledge Verification

Knowledge validation is the final check on the quality of the knowledge graph. This can be done through crowdsourcing verification, expert evaluation, cross validation, user feedback, and other methods. The verified knowledge graphs will be stored in the Knowledge Management Functional Block for further use.

Editor’s note: John will add detail on formal Logic.

#### 6.2.2.5 Knowledge Graph Storage

Verified knowledge graphs are stored in the graph database (Knowledge Management Function Block) for the ENI system to query and use. The Knowledge Management Functional Block stores the constructed knowledge graphs, and responds to requests for knowledge application and knowledge sharing from other functional blocks.

The graph database (GDB) is a database that uses graph structures to store and navigate relationships (i.e., edges) between entities (i.e., nodes) in a graph. A graph database stores nodes and relationships instead of tables, or documents. An edge consists of a start node, end node, type, and direction, and an edge can describe different types of relationships, actions, ownership, and other concepts. There is no limit to the number and kind of relationships a node can have.

There are two popular models of graph databases: property graphs and RDF graphs. Property graphs model relationships between entities, and enable analytics to be performed based on these relationships. RDF graphs conform to a set of W3C standards. An RDF model represents a statement using three elements: two vertices connected by an edge reflecting the subject, predicate and object of a sentence (i.e., an RDF triple). Every vertex and edge is identified by a unique URI, or Unique Resource Identifier. The RDF model provides a way to publish data in a standard format with well-defined semantics, enabling information exchange.

This facilitates executing semantic queries with nodes, edges, and properties. Querying relationships is fast because they are directly stored in the database. Relationships can be intuitively visualized using graph databases, making them useful for heavily inter-connected data.

## 6.3 Using Knowledge Graphs to Manage Intent policies

The ENI system can utilize the knowledge graphs to achieve a series of management functions, including the management of Intent Policies. This clauses describes how it works in the management of Intent Policies.

### 6.3.1 Intent Translation

The translation of an Intent Policy involves taking a source language (e.g., an Intent policy authored in either a restricted language or a DSL) and transforming it into a target language (e.g., a form that the ENI System can understand) and then the low-level network configurations.

Firstly, intent creators express their intent or expected goals for the network in a restricted language or a DSL. The intent policies may be expressed in a high-level and abstract way, e.g., “ensuring high availability”, “optimizing network performance”, or “implementing security policies”. Knowledge graph helps understand these Intents and transform them into the form that ENI system can understand. More specifically, a knowledge graph can refine and understand the Intent of “ensuring high network availability” as a series of indicators related to network availability: Average Minutes of Service, Instantaneous Failure Rate, Load Balancer Accuracy, Network Latency, etc.

The next step is to translate the Intent Policy into low-level network configurations. This step involves mapping the high-level intents to specific device configurations, protocols, and settings. Knowledge graph leverages its structured representation to derive the necessary configurations for network devices. More specifically, when the Intent Policy is to prioritize voice traffic over other data traffic to ensure optimal call quality. The knowledge graph can represent this Intent by specifying policies for traffic classification, quality of service (QoS) settings, and access control lists (ACLs). Then generate device-specific configurations and sent to the Assisted System via API Broker.

Note: ETSI GR ENI 030[i.4] “Transformer Architecture for Policy Translation” provides the method how to the transformer architecture to converts input policies into ENI policies. The detailed procedure of how to use knowledge graph to multi-stage translate Intent Policy will be provided in ETSI GS ENI 033[i.9].

### 6.3.2 Intent Verification

The translated Intent Policy will be verified to ensure that it is executed as expected. In this step, the expected results will be compared with the actual results. The ENI system can use knowledge graphs to verify Intent by checking any discrepancies or deviations between the expected and actual network behaviour. For example, if the Intent Policy is to execute a particular security policy, the knowledge graph can be used to check if the configured access control lists (ACLs) and firewall rules align with the policy.

### 6.3.3 Intent Monitoring

The ENI system continuously monitors the network status after the deployment and execution of Intent policies to ensure the realization of Intent Policies. The ENI system collects data from network devices and systems, and updates the knowledge graphs. Once the changes in network state or user’s Intent is detected that causes the original Intent Policy unable to be met, the Intent creator will be notified in a timely manner.

### 6.3.4 Intent Assurance

When deviations between expected and actual network states are identified, automatic Intent assurance mechanism will be triggered to restore the network to the desired state. Knowledge graph plays a crucial role in determining appropriate remedial measures based on identified issues. It can suggest correcting configurations, re-routing strategies, or adjusting strategies to address network differences. For example, if a network link fails or becomes congested, a knowledge graph can analyse the network topology, determine alternative paths, and recommend rerouting traffic to maintain connectivity and performance. It can also trigger automatic operations to update device configurations, such as disabling or enabling specific interfaces to solve problems.

### 6.3.5 Intent Optimization

Knowledge graph technology contributes to optimizing network policies and recommending best practices. By incorporating domain-specific knowledge and network expertise, the knowledge graph provides insights on network optimization strategies. For example, when a user expresses the intent to enhance network security, the knowledge graph can suggest security measures such as implementing access control lists (ACLs), firewall rules, and intrusion detection systems (IDS) based on known security vulnerabilities and successful security configurations.

Editor’s Note: Link previous knowledge in previous paragraph to 6.4.5 for recommender clarification

## 6.4 Lifecycle Management

### 6.4.1 Lifecycle Management of Intent Policies

#### 6.4.1.1 States of Intent Policy management

Editor’s note: The following text is accepted; however, ideas were raised that need to be considered, for example, is it a conclusion to not show FB Modelling and only states? Also, Extension (the Process of action), Evaluation and monitoring are not in this CR, why? These are examples of questions?

Lifecycle management of Intent Policy is defined and briefly introduced in clause 5.3 of ETSI GR ENI 008 [i.2]. An Intent Policy has its own distinct lifecycle. It is created, translated, verified, deployed, run, and retired. The Intent Creator may request Intent Policy to be created. Once an Intent Policy is created, the Intent Creator may then request that it be enabled or disabled when it is deployed or executed. Finally, the Intent Creator may request that the Intent Policy be removed.

In the present document, the states of Intent Policy are further defined as: **Created**, **Translated**, **Deployed**, **Executed**, **Tested**, **Deleted** and **Terminated**. It’s benefit for ENI system and Intent Creator to define state values, so that they can accurately be aware of the nodes where Intent Policies are processed, and perform appropriate actions. Following are states of Intent Policy and corresponding state values.

* **Created:** The values of Created state include {initialized, in\_process, created, failed}.
* **Translated：**The values of Translated state include {initialized, in\_process, translated, failed\_and\_error, failed\_and\_conflict, failed\_and\_timeout, failed}.
* **Deployed：**The values of Deployed state include {initialized, in\_process, deployed, failed}.
* **Executed:** The values of Executed state include {initialized, in\_process, executed, execution\_aborted, execution\_conflict\_and\_rollback, execution\_conflict\_and\_error, execution\_timeout, failed}.
* **Tested:** The values of Tested state include {initialized, in\_process, tested, testing\_aborted, testing\_timeout, failed}.
* **Deleted:** The values of Deleted state include {initialized, in\_process, deleted\_locally, deleted\_globally, failed}.
* **Terminated:** The values of Terminated state include {Terminated}.

#### 6.4.1.2 State machine of Intent Policy processing

The state machine is able to represent the transition between different states of Intent Policy in an intuitive way. A general high-level state machine of Intent Policy processing is shown as follows.

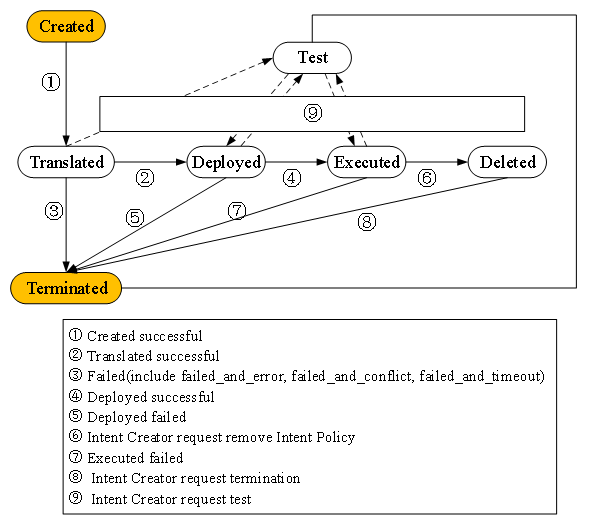


Figure 6-x State transition of intent policy processing

* **Created:** Intent Creator submitted Intent Policy to ENI system. The next state is Translated.
* **Translated:** ENI system translates and transforms the original Intent Policy to a set of recommendations and/or commands that can be executed by the Intent Policy Target. If successful, the Intent Policy will enter the next state (Deployed). If failed, it will be terminated.

NOTE: The failure may be caused by conflict, error, out of scope, etc. ENI system may try to optimize internally (e.g., when encountered with issues) or negotiate with Intent Creator to solve the problems. If ENI system judges the issues can’t be solved within the specified time limit, it will report Intent Creator, who will decide whether to remove the Intent Policy. If the result is that Intent Policy should be removed, the next state is terminated. Deployed state and Executed state are the similar reason, which will not be repeated later.

* **Deployed:** ENI system deploys Intent Policy. If successful, the Intent Policy will enter Executed state. If failed, it will be terminated.
* **Executed:** ENI system executes Intent Policy. If successful, it will wait for the next instruction of Intent Creator, include suspended (not execute), executed, or deleted. If failed, it will be terminated.
* **Deleted:** Deleted state has two options. 1. Deleted locally. Deleting a policy from the active working set of policies but still retaining it in a repository. 2. Deleted globally. Removing a policy from both the active working set as well as all repositories. Both options need to get the permission from Intent Creator.
* **Tested:** Testing is optional. If Intent Creator requests Intent Policy to be tested, ENI system will attempt do so. In addition, at any time after the Created state is entered, the Intent Creator is able to ask ENI System to perform testing. For example, Intent Policy of delay-sensitive scenarios can be directly deployed and executed after translation, and then tested, adjusted, optimized during execution.
* **Terminated:** The end ofIntent Policy lifecycle. The termination of Intent Policy was requested, or some other exception cases occurred, and ENI system has stopped considering the Intent Policy in operation. This is the final clean-up actions. Note that Intent Creator can request to terminate an Intent Policy in any state.

#### 6.4.1.3 Lifecycle Management Roles of Intent Policy

The lifecycle management of Intent Policy is initiated with communication between the Intent Creator and ENI system.

* **Intent Creator**

The Intent Creator is the author of Intent Policy. The Intent Creator has two options for controlling the lifecycle of the Intent Policy, as follows:

1. The first is that, ENI system can only enter the next state or a certain state with the permission of Intent Creator. This is done by requesting the ENI System to perform an appropriate operation on the Intent Policy. For example, if the Intent Creator requests a policy to be created, then the ENI System will attempt to do so. If successful, the Intent Policy will not advance to the next state until the Intent Creator tells the ENI System to do so. It could also proceed to a certain state, at which point it will remain until either the policy fails or the Intent Creator requests a new state to be transitioned to.
2. The second is that, some transitions can be triggered automatically without Intent Creator’s permission. For example, when Intent Policy is translated successfully, it is able to automatically enter the next state (deployed). In this case, Intent Creator only needs to make decisions when ENI system encounters issues, which greatly reduces the workload of Intent Creator.

* **ENI System**

Once an ENI system accepts an Intent Policy, it is obliged to proceed the lifecycle of Intent Policy as well as possible based on the resources and solutions it has available. It is recommended that the ENI System provide status to the Intent Creator.

### 6.4.2 Lifecycle Management of Knowledge Used by Policies

#### 6.4.2.1 Previous Work

NOTE: the text that follows uses the Intent Policy as an example.

In ETSI ENI 008 [i.2], Intent Knowledge refers to the knowledge that is used in the process of Intent Translation. It contains the relevant policyMetadata (e.g. a time period that this intent policy is valid, as well as version information, including a minimum version that can be used) and the generated new knowledge (e.g. word and phrase processing and substitution to translate the original Intent Policy into a form understood by the ENI System for a specific domain). Metadata and knowledge are stored in the model repository and knowledge repository defined in ETSI GS ENI 005 [i.1] within the Repository Management Functional Block, respectively.

The above definition of Intent Knowledge has certain limitations. In this document, Intent knowledge (knowledge used by Intent Policy) is defined as the knowledge used in the whole lifecycle process of Intent Policy processing. It may contain the knowledge generated during Intent Policy translation, testing, conflict detection, as well context knowledge, knowledge from external and so on.

#### 6.4.2.2 Lifecycle Management Roles of Knowledge Used by Policies

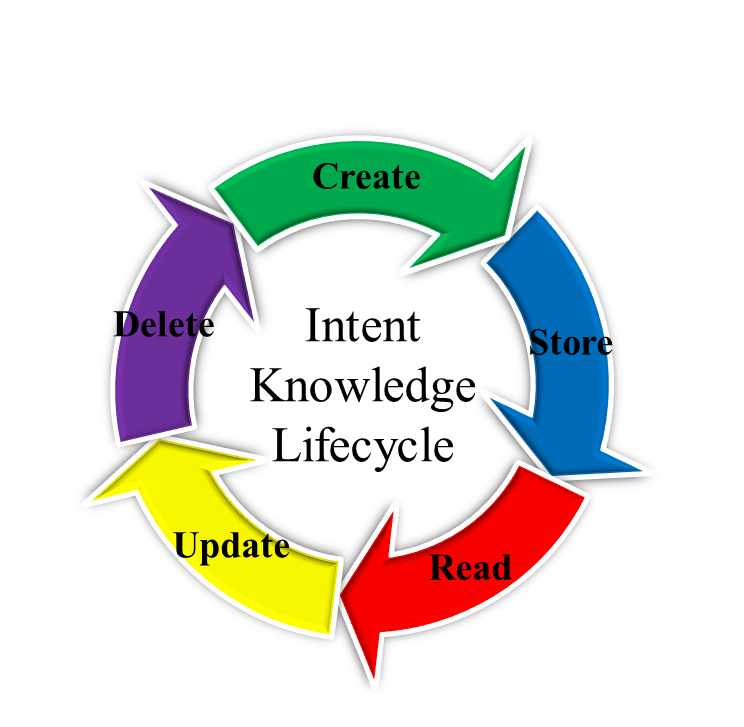
Different from Intent Policy, the lifecycle management of knowledge in this document used by Intent Policy mainly takes place within ENI system (Knowledge Management Functional Block).

The Knowledge Management Functional Block in ENI system is used to store knowledge and generate new knowledge through inferences. Knowledge management works with data, information, knowledge, and wisdom, directing which context and situation information are applied to the raw data, transforming it to information and then knowledge. In addition, this Functional Block defines a formal and consensual representation of knowledge so that computer system could implement the machine learning algorithms and perform reasoning using the knowledge (see clause 6.3.4 of ETSI ENI 005 [i.1]).

#### 6.4.2.3 Lifecycle of Knowledge Used by Policies

Figure 6-4 presents the general lifecycle of knowledge in this document used by Intent Policy. This lifecycle follows the evolution process of CSRUD (Create-Store-Read-Update-Delete). In this figure:

* **Create:** There aremainly two sources of knowledge used by Intent Policy: internally generated knowledge and externally absorbed knowledge. Internally generated knowledge refers to the knowledge created by analyzing policyMetadata and Intent Policy related information, understanding the content of these data and information, then creating knowledge that can be used by Intent Policy. Externally absorbed knowledge may include common sense knowledge and domain specific knowledge. Combination, filtering and fusion may be required for knowledge from different sources.
* **Store:** The knowledgeused by Intent Policy is stored in knowledge repository.
* **Read**: During Intent Policy processing, knowledge will be retrieved for reasoning and decision-making.
* **Update**: Update the existing knowledge repository when new knowledge comes.
* **Delete**: Delete useless and invalid knowledge to reduce storage pressure.

****

**Figure 6-4: Lifecycle of Knowledge Used by Intent Policy**

# 7 Use Cases

## 7.1 Use cases for operators’ business

7.1.1 Use Case #1-1: Intent-driven operating for user-centric cloud-network convergence services

7.1.1.1 Use case context

Cloud services are constantly reshaping social production and lifestyles. Home broadband applications (such as cloud games), large enterprises (such as governmental and finance enterprises), and small- and medium-sized enterprises (such as hospitals and e-commerce enterprises) are greatly facilitated by cloud services. Enterprises are using cloud services to reduce O&M costs, and applications are gradually being migrated to the cloud. The multi-cloud strategy and multi-cloud services have become a trend in the industry. One-hop cloud access via OTN (Optical Transport Network) has been widely used in the industry and has become the mainstream choice for enterprises and cloud leased line services due to its various advantages, including high bandwidth, low latency, hard isolation, high reliability, and one-hop cloud access. However, it has always been difficult to manage services on OTN due to network configuration complexity and multi-vendor collaboration challenges. To overcome the difficulty, the system needs to be more intelligent so that a user without relative knowledge can express his/her demands.

7.1.1.2 Description of the Use Case

7.1.1.2.1 Overview

Cloud private line (CPL) services connect cloud service users to edge or cloud data centres, and edge or cloud data centres to each other, with deterministic connection performance. They may represent point-to-point, point-to-multipoint, multipoint-to-point, or multipoint-to-multipoint connectivity service topologies, and may be implemented using connected or connection-oriented paradigm-supporting technologies. Data centres may be operated by the CPL service customer, by the CPL services provider, by some other service provider(s), or by any combination of these. CPL service traffic consists in machine-to-machine data flows with a range of characteristics. Dynamically mass-customized CPL services are driven operationally by CPL service consumers, either semi-manually (e.g., through a user-facing provisioning portal) or more usefully, directly by consumer scheduling software systems. Intent is obviously a useful service-driving paradigm to support such capability.

7.1.1.2.2 Motivation

In the cloud-network convergence scenario, users need enough bandwidth as well as the computing resources. To be more specific, the user can express an intent of creating a cloud-network convergence service. This intent is then automatically fulfilled by provisioning the corresponding services and allocating the required resources. The already fulfilled intent can be modified by the user. The new intent can be automatically fulfilled by provisioning the corresponding services and allocating the required resources. The Intent-based system monitors the parameters of the cloud-network convergence service (e.g., bandwidth usage), and automatically triggers the closed-loop actions (e.g., increase max bandwidth) in order to guarantee the intent.

The user’s intent, expressed in a natural language, can be translated by the ENI system. The use of the ENI system allows users to achieve their intents only through ordinary descriptions without understanding specific technical details. In this use case, the intent translation process in ENI system is formulated as a question answering(QA) problem, in which the key information can be accurately extracted from the sentences. The ENI system transforms an intent from a simple declarative form to a very complex and technology specific form in an automated way.



**Figure 7-**1 **A General Architecture of Cloud-network Convergence Services**

7.1.1.2.3 Actors and Roles

User: the cloud-network convergence service consumer.

ENI system: receives and translates user’s intent and make closed-loop decisions.

Network Operator: manages the network topology and computing resources.

7.1.1.2.4 Initial context configuration

Both topology of transmission network and information of computing resources have been input into ENI system.

Optimization policy has been input into ENI system.

The natural language processing (NLP) algorithm is well trained and successfully runs in ENI system.

7.1.1.2.5 Triggering conditions

The ENI system receives user’s intent using NLP techniques. For example, a simple user input describing customer’s need in a natural language may be “I need a connection from company A to Cloud Two, with a bandwidth of 2Gbps”.

7.1.1.2.6 Operational flow of actions

1. ENI system receives the user’s intent. The user expresses, in a natural language, an intent to connect one (or more) enterprises to one (or more) clouds, as well as his/her expectation for the quality of the service.
2. ENI system translates the user intent into more specific service intent parameters and sends the recommend configurations to the assisted system.
3. The assisted system receives the recommend configurations to fulfill the user’s intent.
4. The user validates whether the demand is met.
5. ENI system continuously monitors the conditions of the network against the intent specification to ensure compliance with the intent.
6. If the network state cannot meet the user’s intent, ENI system sends new recommend configurations to the assisted system.

7.1.1.2.7 Post-conditions

The intent translation and intent life-cycle management have been achieved.

The intent assurance is achieved by the closed-loop automation.

## 7.2 Use cases for vertical industry

7.2.1 Use Case #2-1: Intent-Driven Home Intranet Management

7.2.1.1 Use case context

Home router hardware has been upgraded for generations. Currently, the hardware performance of a home router is capable of more complex jobs, other than routing. Meanwhile, the latest router system normally is compatible with complex jobs, such as workload balancing, resource orchestration, etc. However, to accomplish these jobs mentioned above, it is requested that the user must have some computer and network management basics for now. This pre-requisite has become a barrier to making the home Intranet develop further. To overcome the difficulty, the system needs to be more intelligent so that a user without relative knowledge can express his/her demands and the router responds correctly.

With the vigorous development of the Internet of Things and the emergence of various intelligent devices, the concept of ‘Smart Home’ has gradually become the focus of the industry. One of the core ideas of Smart Home is to manage the family Intranet according to user intent policies.

7.2.1.2 Description of the Use Case

7.2.1.2.1 Overview

The access of many smart devices to the home network brings many new services such as Extended Reality (XR), Network Attached Storage(NAS), Ultra High Definition(UHD) video streaming, which gradually complicates the structure of the home network and requires more dedicated network management.

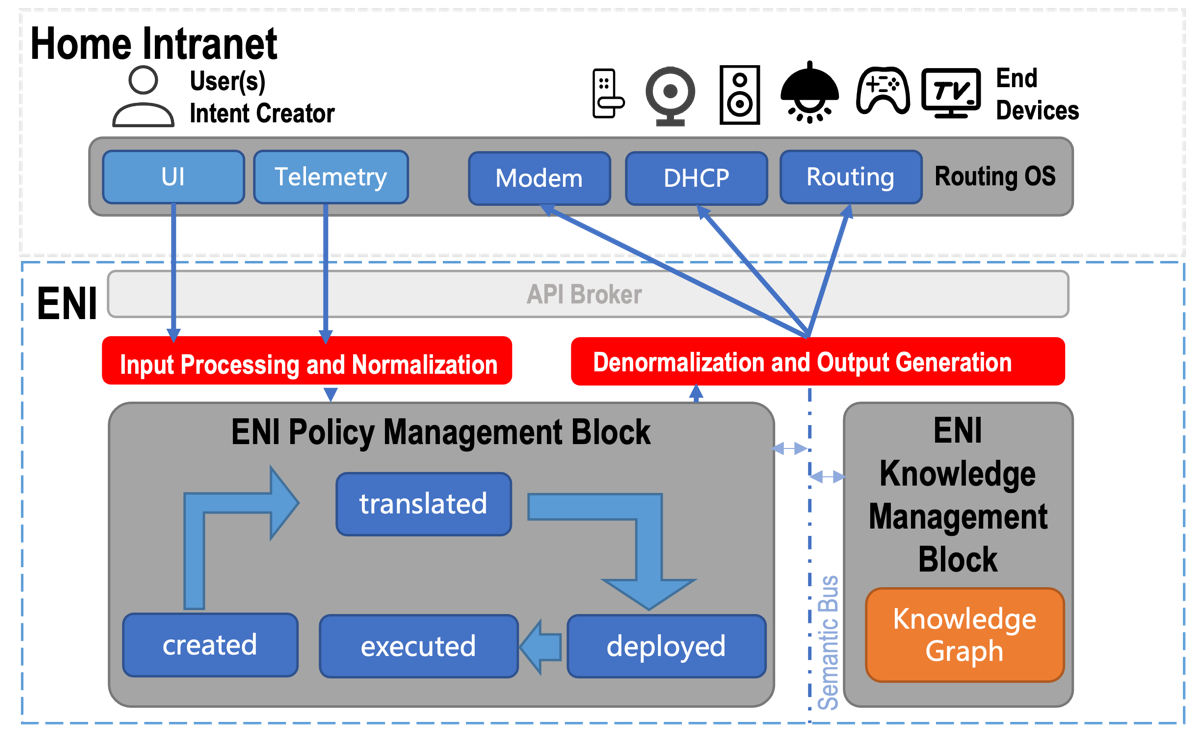
Normally, network management is a very complex process for a home user, which takes a considerable amount of time for learning the relative knowledge To accomplish family Intranet management, a user is requested to:

* Have basic network knowledge. The user must get familiar with common network protocols and packet transmission mechanisms. The user must be able to come up with a reasonable management plan.
* Be familiar with Network Management. The user needs to have the experience on managing network so that the user can understand what different indexes of the network mean, which kind of data should be paid more attention, etc.

7.2.1.2.2 Motivation

The current home network management method is single. The user must input network policy directly to the network. Thus, there are certain requirements for users' professional skills. However, home users often do not have much experience on network management. Moreover, different from other network management scenarios, since home network users are usually short of relative knowledge, the user's intention input is usually more obscure than that of other networks. For example, these users describe requirements for the service layer only. To achieve the request, the network needs to orchestrate the resources.

ENI system can translate the intent policy input in the form of restricted natural language. The introduction of ENI system in the home network environment can effectively reduce the knowledge requirement threshold for users to manage the network. With the help of ENI system, users can not only use traditional computer language to operate the network, but also express their expectations for the network or a specific business running on the network, by directly using natural language. ENI system can accurately extract the keywords in the sentences, dig out the real needs of users, translate the needs into appropriate network commands, and automatically dispatch them to the corresponding home network devices.



**Figure 7-2 A General Architecture of Home Intranet Management**

7.2.1.2.3 Actors and Roles

User: the family Intranet user. We assume in this case, the user has not learned any relevant skills.

End Devices: Intelligent devices connected to the family Intranet. Normally, they are laptops, intelligent TVs, cell phones, VR helmets, etc.

ENI System: The system that receives intent polices from the users, and translates the intent policies into polices.

Running Applications: Applications that are currently running on any end device.

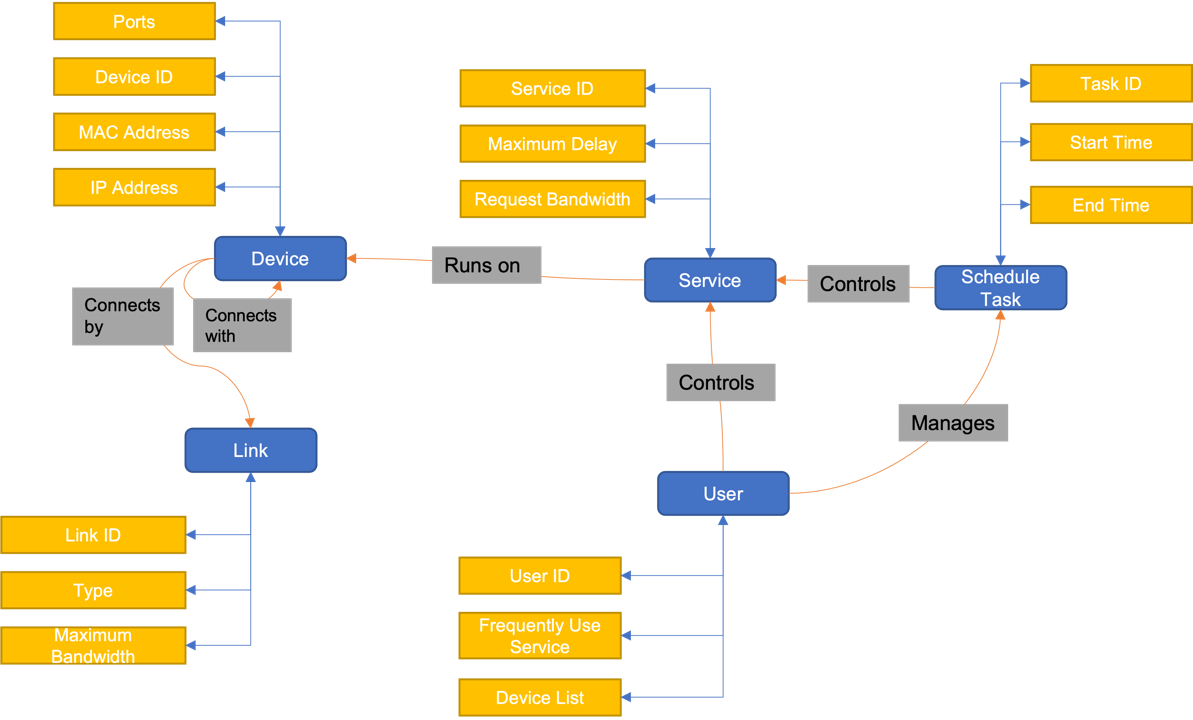
Terminated Applications: Applications that were running on end devices but terminated before ENI system starts processing new intent policy.

7.2.1.2.4 Initial context configuration

End Devices are connected to the home Intranet and they are all functional.

The home Intranet runs in a stable state.

ENI constructs a knowledge graph with the following attributes and relations and stores in knowledge management functional block:



**Figure 7-3 Ontology/schema structure of the Home Intranet Management**

7.2.1.2.5 Triggering conditions

The user inputs an intent policy which implies a state the user expects the network or running business to achieve.

7.2.1.2.6 Operational flow of actions

1. ENI system receives the new intent policy from users or running applications. Then ENI system validates if there are any conflicts among new intent policy and processed intent polices. If there is any, jump to step 2a). Otherwise, continue to step 2b).

2a) If the processed Intent Policies were from terminated applications, continue to 2b). Otherwise, jump to 5).

2b) ENI system translates the new intent policy into suitable recommendations or commands with the assistance of the knowledge graph. And then ENI dispatches the polices to home router.

3) Home router receives the polices and modify the resource allocation plan which leads the state of the home Intranet changes.

4) End devices modify the resource consuming strategy to accommodate the change of network.

5) ENI system writes the information down into a log and reports it to the Routing OS along with recommendations and commands.

6) The user validates whether the application’s or user’s demand is met.

7） If not, the user may input new intent policy in form of natural language or computer programming language to correct the previous intent policy.

The step 7) is optional. If the network state meets the user demand, the step should be skipped.

7.2.1.2.7 Post-conditions

The state of the home Intranet or a business has been changed accordingly.

The user’s intent policy has been achieved.

# 8 Conclusions

Annex :  
Change History

| Date | Version | Information about changes |
| --- | --- | --- |
| <Month year> | <#> | <Changes made are listed in this cell> |
|  |  |  |
|  |  |  |
|  |  |  |

# History

|  |  |  |
| --- | --- | --- |
| **Document history** | | |
| <Version> | <Date> | <Milestone> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Latest changes made on 2022-03-14*