ETSI GR ENI 008 V2.1.1 (2021-03)

Experiential Networked Intelligence (ENI); InTent Aware Network Autonomicity (ITANA)

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.

Reference
DGR/ENI-0013_Net_Autonomicity

Keywords
artificial intelligence, management

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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

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

Copyright Notification

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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/).

Pursuant to the ETSI IPR Policy, no investigation, 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.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Experiential Networked Intelligence (ENI).

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 (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 will discuss various design options, in terms of a set of new stand-alone and/or nested Functional Blocks, for using intent with the ENI System Architecture. This includes accepting, translating and validating intent statements, determining how intent affects the goals and operation of the ENI system, and how it is used by business users, application developers and network administrators.

The present document provides a set of recommendations and conclusions whose main scope is ETSI GS ENI 005 [i.2] Release 2. However, the contents of the present document also affect other GSs & GRs (e.g. ETSI GS ENI 001 [i.1], ETSI GS ENI 002 [i.4] and ETSI GR ENI 004 [i.3]).

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 001 (V3.1.1): "Experiential Networked Intelligence (ENI); ENI use cases".

[i.2] ETSI GS ENI 005 (V2.1.1): "Experiential Networked Intelligence (ENI); System Architecture".

[i.3] ETSI GR ENI 004 (V3.1.1): "Experiential Networked Intelligence (ENI); Terminology for Main Concepts in ENI".

[i.4] ETSI GS ENI 002 (V3.1.1): "Experiential Networked Intelligence (ENI); ENI requirements".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

compiler: computer program that translates computer code written in one programming language (the source language) into another language (the target language) (see ETSI GS ENI 005 [i.2])

NOTE: The term "compiler" is primarily used for programs that translate source code from a high-level programming language to a lower level language.

intent creator: set of authorized entities that is able to create an Intent Policy including the User, APP, OSS, BSS and Orchestrator

NOTE: The Intent Creator submits the Intent Policy to the ENI system through dedicated External Reference Points.

intent policy target: entity whose behaviour is affected by the Intent Policy

NOTE: For the purposes of the present document, this is either the Assisted System (or its Designated Entity) or the ENI System itself.

interpreter: computer program that directly executes instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program (see ETSI GS ENI 005 [i.2])

policyMetadata: data that describes and prescribes policy characteristics and behaviour (see ETSI GS ENI 005 [i.2])

NOTE: Examples of policyMetadata include a time period that this intent policy is valid, as well as version information, including a minimum version that will be used.

3.2 Symbols

Void.

3.3 Abbreviations

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

API	Application Programming Interface
APP	Application (Functional Block)
BS	Base Station
BSS	Business Support Systems
CRUD	Create Read Update Delete
DPI	Deep Packet Inspection
DSL	Domain-Specific Language
EMS	Element Management System
ENI	Experiential Networked Intelligence
EPC	Evolved Packet Core
FB	Functional Block
IDMS	Intent-Driven Management System
IMS	IP Multimedia Subsystem
IP	Internet Protocol
ITANA	InTent Aware Network Autonomicity
NFV	Network Functions Virtualisation
NR	New Radio
OAM	Operating and Maintenance
OPEX	OPeration EXpediture
OSS	Operations Support System
RAN	Radio Access Network
SDO	Standard Developing Organization
SFC	Service Function Chain
SLA	Service Level Agreement
SON	Self-Organizing Network
UID	Unique IDentifier
VNF	Virtual Network Function

4 Introduction

Intent Policy is introduced and defined in clause 6.3.9.3.2 of ETSI GS ENI 005 [i.2], which uses a restricted natural language (e.g. external DSL) to express the goals of the policy, without describing how to accomplish the goals. The Intent Policy concept could be applied in various network domains, such as a wireless network and a data centre.

Intent policies enable different users of the ENI System to author policies in a form that a particular constituency of users understands. For example, business users are able to write Intent Policies using business terms without having to use a language that they do not normally use, such as a general-purpose programming language like Java or Python.

The Policy Continuum (see clause 6.3.9.6.2 of ETSI GS ENI 005 [i.2]) is used to formally differentiate between the needs of different constituencies in defining and expressing policy. Each constituency defines a common set of concepts and terminology. Hence, an Intent Policy Rule for one constituency may be different in structure and content than an Intent Policy Rule for a different constituency. More importantly, since an Intent statement does not specify how it should be implemented, every intent statement needs to be translated to a more concrete form for validation and processing. For example, Business needs are generated from many types of users, such as business users and application developers. Hence, the ENI system uses the Policy Continuum to represent the different constituencies served and to translate an intent policy at a higher level of abstraction to a policy at a lower level of translation. Such a translation is performed one or more times (e.g. to go from a business view to a system view to an administrator view).

5 Impact of Intent Policies on the System Architecture

5.1 Architecture Enhancement for Intent Policies

5.1.1 Scope of Intent Policies

There are two different ways that Intent Policies can be used:

1) An External Entity (e.g. an Operator) sends an Intent Policy to the ENI System that affects the behaviour of the Assisted System (or its Designated Entity).

2) An External Entity sends a Policy (of any type) to the ENI System that affects the behaviour of the ENI System.

In each case, the External Entity sends an Intent Policy. The ENI System will translate the Intent Policy, process it, and then produces a set of recommendations and/or commands that realize the Intent Policy. This is done by the Policy Management Functional Block of the ENI System. The difference is the target of the Intent Policy:

1) If the target is the *Assisted System* (or its Designated Entity), then the set of recommendations and/or commands produced are sent to the Denormalisation and Output Generation Functional Blocks, where they are denormalised and formatted. They are then sent to the API Broker, which transmits them to the Assisted System (or its Designated Entity).

2) If the target is the *ENI System*, then the set of recommendations and/or commands produced are sent to the set of affected Functional Blocks of the ENI System.

5.1.2 Adding an Intent Translation Functional Block

A new Functional Block named Intent Translation is introduced to support the use of an Intent Policy process that includes intent translation, intent assurance and the lifecycle management of Intent Policy.

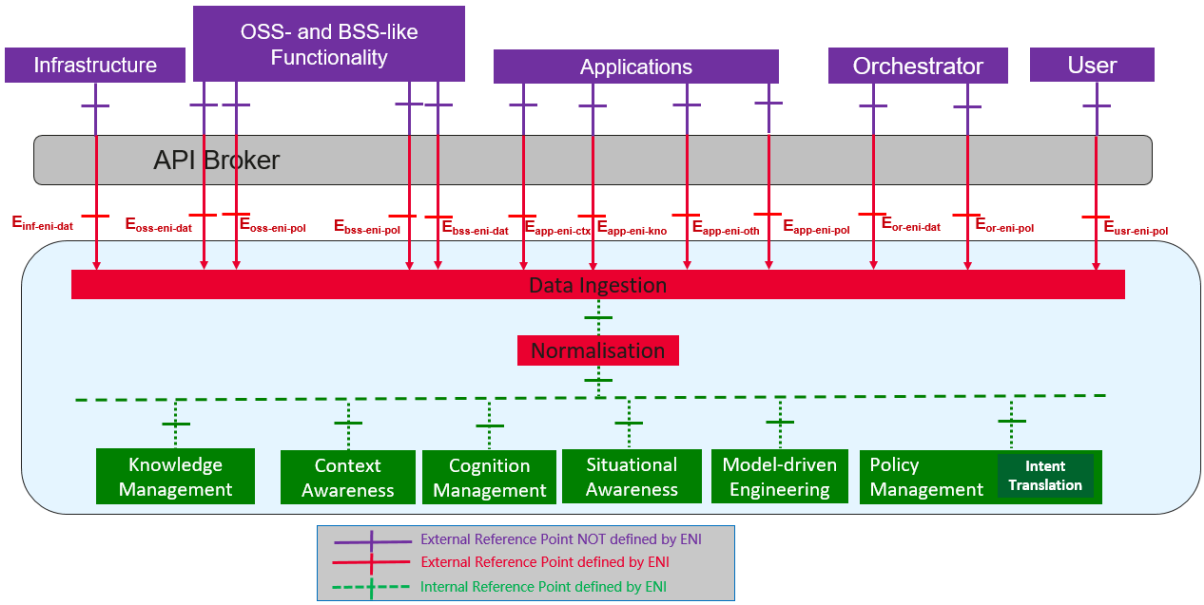

Figure 5-1: Architecture Enhancement for Intent Policy with its Input Reference Points

In this context, the following definitions and considerations apply:

- **Intent Translation Functional Block:** a Functional Block recommended to be added within the Policy Management Functional Block that takes part in the process of Intent Translation. It performs lexical analysis, syntactic analysis, semantic analysis and augmentation, either parsing or compiling and, optionally, interpretation of the Intent Policy.

- **Process of Intent Translation:** the procedure to translate the Intent Policy to the desired format executed internally by the ENI System or transformed into recommendations/commands sent to the Assisted System. This process is discussed in more detail in clause 5.2.

- **Process of Intent Assurance:** the procedure to maintain and monitor the execution of an Intent Policy, including detecting and reporting any conflicts and failures. The use of statistics and possibly analytics to summarize these aspects of Policy will ensure that the requirements of the Intent Creator (e.g. User/APP/OSS-/and BSS-like Functionality), see clause 5.2.2 regarding its definition, are satisfied. This process is discussed in more detail in clause 5.3.

- **Operational management of Intent Policy:** Create Read Update Delete (CRUD) commands are operations that affect the state of Intent Policies. Intent policies are sent from an External Entity or the Intent Creator (see clause 5.4 for more information) to the ENI System. CRUD Intent Policies are used to manage the Assisted System (or its Designated Entity) or the ENI System itself.

- **Intent knowledge:** 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.2] within the Repository Management Functional Block, respectively.

NOTE: The original form of an Intent Policy uses a restricted natural language. This is likely to contain both ambiguities in the meaning of the Intent Policy as well as terms familiar to the Intent Creator that need to be translated to a form that can be understood by the entities affected by this Intent Policy. This is done by the ENI System. Recommendations and/or commands are produced by the Model-Driven Engineering FB, which are then packaged as ENI Policies by the Policy Management FB. If the target of the Intent Policy was the ENI System itself, then no further processing is required, and the ENI System will execute those recommendations and/or commands. If the target of the Intent Policy was the Assisted System (or its Designated Entity), then the transformed Intent Policy is denormalised and formatted, and then sent to the API Broker. The API Broker sends the Intent Policy to the Assisted System (or its Designated Entity).

5.1.3 Reference Points for Intent Policies Implementation

The internal reference points involved in the Intent Policies process are shown in Figure 5-2.

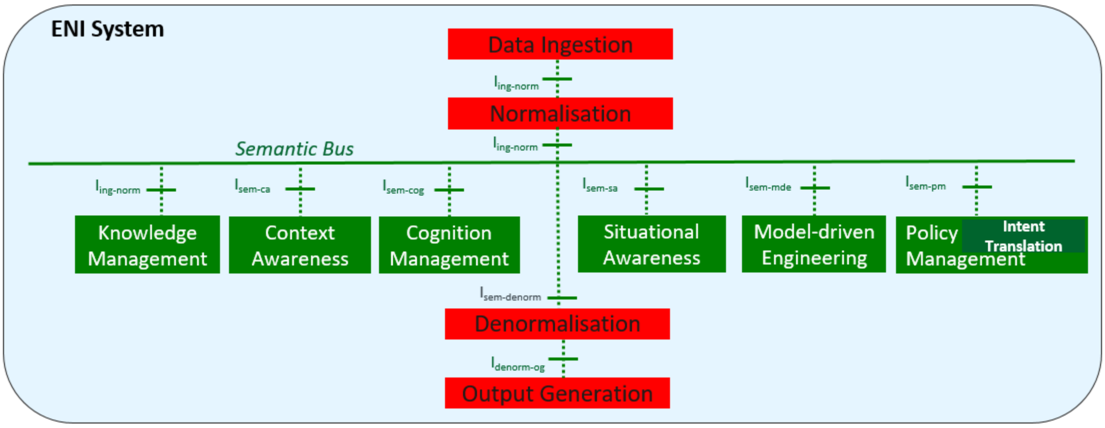

Figure 5-2: Overview of the ENI Internal Reference Points

Table 5-1, depicted further down in clause 5.2.3.2, provides brief descriptions of the Intent Policy and associated information and/or metadata exchanged during the translation process.

5.2 Translation of Intent Policies

5.2.1 Introduction

Intent Policy is first translated to an intermediate form (a.k.a intermediate representation), which is typically a set of data structures (and possibly code) that is used to further analyse and transform the original entry. For example, human-readable text could be transformed into a graph. This type of processing is performed by the Intent Translation Functional Block, as shown in Figure 5-3. The translation process possibly will use more than one intermediate forms (e.g. if more than one intent abstraction level needs to be processed).

The second step is to further analyse the Intent Policy, both syntactically and semantically. At this stage, the Intent Policy is modified to correct errors, remove ambiguities, and transformed into a form that is more conducive for changing the intermediate form into recommendations and/or commands that the Policy Intent Target is able to understand. At this point, the Intent Policy is ready to be executed. The data processing may then continue recursively with Functional Blocks interacting with each other.

In the existing operation of a network, Intent Policy translation is done by experienced operational staff. The use of this knowledge in an ENI system for Intent Policy translation is for further study.

5.2.2 Intent Policy Translation Functional Block Architecture

The Intent Translation Functional Block is a part of the Policy Management Functional Block, and is responsible for translating and transforming the original Intent Policy submitted to a set of recommendations and/or commands that can be applied to the Intent Policy Target. Figure 5-3 shows the Functional Blocks that make up the Intent Translation Functional Block, as shown inside the dotted light blue rectangle. The outer green rectangle shows the Policy Management Function Block (which contains the Intent Translation Functional Block). The outer dashed rectangle denotes the ENI System.

The Intent Translation Functional Block includes the Intent Parser, Syntax Analyser, Semantic Analyser, Local Conflict Resolution, Intent Abstraction Translator and Intent Policy Compiler. The Policy Decision Engine, Policy Execution Engine, and Policy Verification Engine are embedded Functional Blocks that perform runtime processing of ENI Policies within the Policy Management Functional Block. The following parts of this clause will introduce the roles and tasks allocated to each Functional Block, and then a detailed interaction of flows with other ENI Functional Blocks will be introduced in clause 5.2.3.

Figure 5-3 shows the involved Functional Blocks to translate Intent Policies. The Functional Blocks interactions inside the black-dotted box illustrate the process to translate Intent Policies inside the Policy Management Function Block. The blue box illustrates the import Functional Blocks that are required for the processing of Intent Policies, which consists of the nested Intent Translation Functional Block.

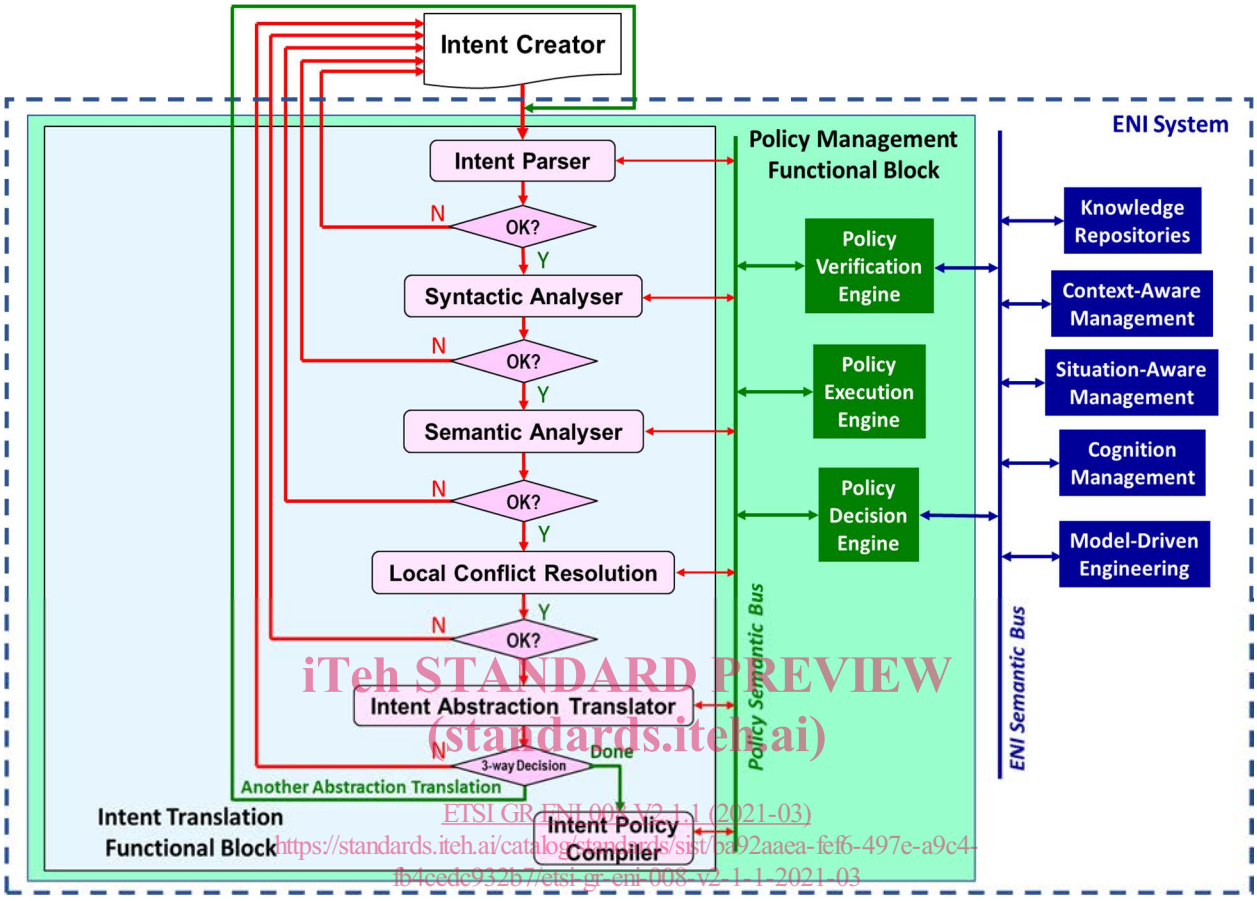

Figure 5-3: Translation of Intent Policies

Intent Creator: This is a set of authorized entities including the User/APP/OSS/BSS and Orchestrator external entities that is able to create an Intent Policy. The Intent Creator submits the Intent Policy to the ENI System through dedicated External Reference Points.

Intent Parser: The Intent Parser is responsible for the initial parsing of the submitted Intent Policy. This typically consists of lexical analysis, token generation, and syntactic analysis (though other functions can also be included). Lexical analysis is the first phase, and takes an input (pre-processed to be a sentence), compares it to a grammar, and then breaks the sentence into sets of characters. The next phase generates tokens from the set of characters that are defined using the grammar rules of the Intent Policy language (e.g. the identification of nouns and verbs). The final phase is syntactical analysis, which ensures that the analysed sentence is grammatically correct. For example, the sentence "Send Host Computer to the Notifications" contains legal tokens, but is grammatically incorrect (it should be "Send Notifications to the Host Computer"). The output of the Syntactical Analyser is either a Concrete or an Abstract Syntax Tree. If any error happens during the parsing process, appropriate error messages are sent to the Intent Creator, and the Knowledge Management Functional Block records the errors for further analysis.

Semantic Analyser: The purpose of the Semantic Analyser is to provide meaning to the output of the Syntax Analyser. Examples include datatype checking, array bounds checking, proper declaration of variables, and scope resolution. For example, the expression "int x = "aValue" will pass lexical and syntactic analysis, as it is structurally correct. However, it will generate a semantic analysis error, since the datatype of the variable does not match the datatype of the value. The Semantic Analyser in ENI generates gists and keywords to provide additional knowledge to other Functional Blocks. If any error happens during this process, error messages are sent to the Intent Creator, and the Knowledge Management Functional Block records the errors for further analysis.

Local Conflict Resolution: This module interacts with the Knowledge Repository and checks if the current Intent Policy conflicts with any existing Policies. For the purposes of the present document, a *policy conflict* is defined as two policies that, when executed, cause contradictory and otherwise incompatible results within a given policy execution time window. For example, if Intent Policy 1 sets the value of an attribute named numErrors to "2" at 08:00:00, and intent Policy 2 then sets the value of the numErrors attribute to 3 at 08:00:01, that is a policy conflict. If any conflict is detected during this process, conflict messages and error information are sent to the Intent Creator, and Knowledge Management Functional Block records the errors for further analysis.

NOTE 1: While the above definition of a policy conflict applies to imperative policies, there are additional problems for intent policies. This is for further study.

Intent Abstraction Translator: The Intent Abstraction Translator retrieves knowledge from the Knowledge Management FB to identify the abstraction level of the input Intent Policy. It then defines the target output abstraction level, as well as the types of output configuration parameters, based on the semantics of the Intent Policy, the gist and keywords, and the input abstraction level. Other information, including Intent Policy metadata, could also be used to help to identify the abstraction levels. The abstraction level refers to the different views of Policy Continuum [i.2]. For instance, an end user authors an Intent Policy to improve video streaming quality without supplying any technical details The Intent Abstraction Translator identifies the abstraction level of the input Intent Policy as the business view level. Similarly, in order to be enforced, the target output abstraction level needs to be at the Instance view level. It is up to the Intent Translation Functional Block to decide if one or more translations are required between the input and output abstraction levels. Once the current abstraction level has successfully completed, this module then determines if another abstraction level translation is required, or if it is finished. If the former, control returns to the Intent parser with new instructions; if the latter, the completed Intent Policy is sent to the Intent Policy Compiler.

NOTE 2: An alternative would be for the Intent Translation Functional Block to simply map this intent to a known SLA, such as Gold or Platinum Service.

Intent Policy Compiler: The Intent Policy Compiler compiles the finalized Intent Policy output into a form that can be processed by the Model Driven Engineering Functional Block.

All other Functional Blocks are defined in ETSI GS ENI 005 [i.2] where the policy continuum is specified.

5.2.3 Procedure for Translating DSL Intent Policies

5.2.3.1 Introduction

This clause describes the procedure for intent policy translation when the intent policy is expressed by a DSL, and it is based on the architecture described in clause 5.1.

5.2.3.2 Translation Overview

Figure 5-4 is a simplified message sequence diagram that illustrates the steps between the visible actors (i.e. the entity that authors the policy, the ENI System Functional Blocks, and the Assisted System, thus excluding the ENI System Internal Functional Blocks as well as the ENI System Internal Interfaces). These steps help to describe the Intent Translation process. Figure 5-5, depicted further down, consists of a more detailed message flow sequence diagram where interaction amongst all involved ENI Functional Blocks is shown.