

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: May 7, 2020

A. Clemm  
Futurewei  
L. Ciavaglia  
Nokia  
L. Granville  
Federal University of Rio Grande do Sul (UFRGS)  
J. Tantsura  
Apstra, Inc.  
November 4, 2019

**Intent-Based Networking - Concepts and Overview**  
**draft-clemm-nmrg-dist-intent-03**

**Abstract**

Intent and Intent-Based Networking are taking the industry by storm. At the same time, those terms are used loosely and often inconsistently, in many cases overlapping and confused with other concepts such as "policy". This document is intended to clarify the concept of "Intent" and provide an overview of functionality that associated with it. The goal is to contribute towards a common and shared understanding of terms, concepts, and functionality which can be used as foundation to guide further definition of associated research and engineering problems and their solutions.

**Status of This Memo**

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 7, 2020.

**Copyright Notice**

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">2</a>
<a href="#">2.</a>	Key Words . . . . .	<a href="#">4</a>
<a href="#">3.</a>	Definitions and Acronyms . . . . .	<a href="#">4</a>
<a href="#">4.</a>	Introduction of Concepts . . . . .	<a href="#">5</a>
<a href="#">4.1.</a>	Intent and Intent-Based Management . . . . .	<a href="#">5</a>
<a href="#">4.2.</a>	Related Concepts . . . . .	<a href="#">6</a>
<a href="#">4.2.1.</a>	Service Models . . . . .	<a href="#">6</a>
<a href="#">4.2.2.</a>	Policy and Policy-Based Management . . . . .	<a href="#">8</a>
4.2.3.	Distinguishing between Intent, Policy, and Service Models . . . . .	<a href="#">10</a>
<a href="#">5.</a>	Principles . . . . .	<a href="#">11</a>
<a href="#">6.</a>	Lifecycle . . . . .	<a href="#">14</a>
<a href="#">7.</a>	Intent-Based Networking - Functionality . . . . .	<a href="#">18</a>
<a href="#">7.1.</a>	Intent Fulfillment . . . . .	<a href="#">18</a>
<a href="#">7.2.</a>	Intent Assurance . . . . .	<a href="#">18</a>
<a href="#">8.</a>	Items for Discussion . . . . .	<a href="#">19</a>
<a href="#">9.</a>	IANA Considerations . . . . .	<a href="#">19</a>
<a href="#">10.</a>	Security Considerations . . . . .	<a href="#">19</a>
<a href="#">11.</a>	References . . . . .	<a href="#">19</a>
<a href="#">11.1.</a>	Normative References . . . . .	<a href="#">19</a>
<a href="#">11.2.</a>	Informative References . . . . .	<a href="#">19</a>
	Authors' Addresses . . . . .	<a href="#">21</a>

## [1.](#) Introduction

Traditionally in the IETF, interest with regard to management and operations has focused on individual network and device features. Standardization emphasis has generally been put on management instrumentation that needed to be provided to a networking device. A prime example for this is SNMP-based management and the 200+ MIBs that have been defined by the IETF over the years. More recent examples include YANG data model definitions for aspects such as interface configuration, ACL configuration, or Syslog configuration.

There is a sense and reality that in modern network environments managing networks by configuring myriads of "nerd knobs" on a device-



by-device basis is no longer sustainable. Big challenges arise with keeping device configurations not only consistent across a network, but consistent with the needs of services and service features they are supposed to enable. Adoptability to changes at scale is a fundamental property of a well designed IBN system, that requires ability to consume and process analytics that are context/intent aware at near real time speeds. At the same time, operations need to be streamlined and automated wherever possible to not only lower operational expenses, but allow for rapid reconfiguration of networks at sub-second time scales and to ensure networks are delivering their functionality as expected.

Accordingly, IETF has begun to address end-to-end management aspects that go beyond the realm of individual devices in isolation. Examples include the definition of YANG models for network topology [[RFC8345](#)] or the introduction of service models used by service orchestration systems and controllers [[RFC8309](#)]. In addition, a lot of interest has been fueled by the discussion about how to manage autonomic networks as discussed in the ANIMA working group. Autonomic networks are driven by the desire to lower operational expenses and make management of the network as a whole exceptionally easy, putting it at odds with the need to manage the network one device and one feature at a time. However, while autonomic networks are intended to exhibit "self-management" properties, they still require input from an operator or outside system to provide operational guidance and information about the goals, purposes, and service instances that the network is to serve.

This vision has since caught on with the industry in a big way, leading to a significant number solutions that offer "intent-based management" that promise network providers to manage networks holistically at a higher level of abstraction and as a system that happens to consist of interconnected components, as opposed to a set of independent devices (that happen to be interconnected). Those offerings include IBN systems (offering full lifecycle of intent), SDN controllers (offering a single point of control and administration for a network) as well as network management and Operations Support Systems (OSS).

However, it has been recognized for a long time that comprehensive management solutions cannot operate only at the level of individual devices and low-level configurations. In this sense, the vision of "intent" is not entirely new. In the past, ITU-T's model of a Telecommunications Management Network, TMN, introduced a set of management layers that defined a management hierarchy, consisting of network element, network, service, and business management. High-level operational objectives would propagate in top-down fashion from upper to lower layers. The associated abstraction hierarchy was key



to decompose management complexity into separate areas of concerns. This abstraction hierarchy was accompanied by an information hierarchy that concerned itself at the lowest level with device-specific information, but that would, at higher layers, include, for example, end-to-end service instances. Similarly, the concept of "policy-based management" has for a long time touted the ability to allow users to manage networks by specifying high-level management policies, with policy systems automatically "rendering" those policies, i.e. breaking them down into low-level configurations and control logic.

What has been missing, however, is putting these concepts into a more current context and updating it to account for current technology trends. This document attempts to clarify the concepts behind intent. It differentiates it from related concepts. It also provides an overview of first-order principles of Intent-Based Networking as well as associated functionality. In addition, a number of research challenges are highlighted. The goal is to contribute to a common and shared understanding that can be used as a foundation to articulate research and engineering problems in the area of Intent-Based Networking.

## **2. Key Words**

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#) [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

## **3. Definitions and Acronyms**

ACL: Access Control List

Intent: An abstracted, declarative and vendor agnostic set of rules used to provide full lifecycle (Design/Build/Deploy/Validate) to a network and services it provides.

Policy: A rule, or set of rules, that governs the choices in behavior of a system.

SSoT: Single Source of Truth - A functional block in an IBN system that normalizes user' intent and serves as the single source of data for the lower layers.

IBA: Intent Based Analytics - Analytics that are defined and derived from user' intent and used to validate the intended state.



IBS: Intent Based System.

PDP: Policy Decision Point

PEP: Policy Enforcement Point

Service Model: A model that represents a service that is provided by a network to a user.

## **4. Introduction of Concepts**

The following section provides an overview of the concept of intent respectively intent-based management. It also provides an overview of the related concepts of service models, and of policies respectively policy-based management, and explains how they relate to intent and intent-based management.

### **4.1. Intent and Intent-Based Management**

In the context of Autonomic Networks, Intent is defined as "an abstract, high-level policy used to operate a network" [[RFC7575](#)]. According to this definition, an intent is a specific type of policy. However, to avoid using "intent" simply as a synonym for "policy, a clearer distinction needs to be introduced that distinguishes intent clearly from other types of policies.

For one, while Intent-Based Management clearly aims to lead towards networks that are dramatically simpler to manage and operate requiring only minimal outside intervention, the concept of "intent" is not limited to autonomic networks, but applies to any network. Networks, even when considered "autonomic", are not clairvoyant and have no way of automatically knowing particular operational goals nor what instances of networking services to support. In other words, they do not know what the "intent" of the network provider is that gives the network the purpose of its being. This still needs to be communicated by what informally constitutes "intent".

More specifically, intent is a declaration of operational goals that a network should meet and outcomes that the network is supposed to deliver, without specifying how to achieve them. Those goals and outcomes are defined in a manner that is purely declarative - they specify what to accomplish, not how to achieve it. "Intent" thus applies several important concepts simultaneously:

- o It provides data abstraction: Users and operators do not need to be concerned with low-level device configuration and nerd knobs.





- o It provides functional abstraction from particular management and control logic: Users and operators do not need to be concerned even with how to achieve a given intent. What is specified is a desired outcome, with the intent-based system automatically figuring out a course of action (e.g. a set of rules, an algorithm) for how to achieve the outcome.

In an autonomic network, intent should be rendered by the network itself, i.e. translated into device-specific rules and courses of action. Ideally, it should not even be orchestrated or broken down by a higher-level, centralized system, but by the network devices themselves using a combination of distributed algorithms and local device abstraction. Because intent holds for the network as a whole, not individual devices, it needs to be automatically disseminated across all devices in the network, which can themselves decide whether they need to act on it. This facilitates management even further, since it obviates the need for a higher-layer system to break down and decompose higher-level intent, and because there is no need to even discover and maintain an inventory of the network to be able to manage it.

Tentative definition for intent-based networks Networks configuring and adapting autonomously to the user or operator intentions (i.e., a desired state or behavior) without the need to specify every technical detail of the process and operations to achieve it (i.e., the "machines" will figure out on their own how to realize the user goal).

Other definitions of intent exist such as [[TR523](#)] and will be investigated in future revisions of this document. Likewise, some definitions of intent allow for the presence of a centralized function that renders the intent into lower-level policies or instructions and orchestrates them across the network. While to the end user the concept of "intent" appears the same regardless of its method of rendering, this interpretation opens a slippery slope of how to clearly distinguish "intent" from other higher-layer abstractions. Again, these notions will be further investigated in future revisions of this document and in collaboration with NMRG.

## **[4.2.](#) Related Concepts**

### **[4.2.1.](#) Service Models**

A service model is a model that represents a service that is provided by a network to a user. Per [[RFC8309](#)], a service model describes a service and its parameters in a portable/vendor agnostic way that can be used independent of the equipment and operating environment on which the service is realized. Two subcategories are distinguished:



a "Customer Service Model" describes an instance of a service as provided to a customer, possibly associated with a service order. A "Service Delivery Model" describes how a service is instantiated over existing networking infrastructure.

An example of a service could be a Layer 3 VPN service [[RFC8299](#)], a Network Slice, or residential Internet access. Service models represent service instances as entities in their own right. Services have their own parameters, actions, and lifecycles. Typically, service instances can be bound to end users, who might be billed for the service.

Instantiating a service typically involves multiple aspects:

- o A user (or northbound system) needs to define and/or request a service to be instantiated.
- o Resources need to be allocated, such as IP addresses, AS numbers, VLAN or VXLAN pools, interfaces, bandwidth, or memory.
- o How to map services to the resources needs to be defined. Multiple mappings are often possible, which to select may depend on context (such as which type of access is available to connect the end user with the service).
- o [[I-D.ietf-teas-te-service-mapping-yang](#)] is an example of such mapping - a data model to map customer service models (e.g., the L3VPM Service Model) to Traffic Engineering (TE) models (e.g., the TE Tunnel or the Abstraction and Control of Traffic Engineered Networks Virtual Network model)
- o Bindings need to be maintained between upper and lower-level objects.
- o Once instantiated, the service needs to be validated and assured to ensure that the network indeed delivers the service as requested.

They involve a system, such as a controller, that provides provisioning logic. Orchestration itself is generally conducted using a "push" model, in which the controller/manager initiates the operations as required, pushing down the specific configurations to the device. (In addition to instantiating and creating new instances of a service, updating, modifying, and decommissioning services need to be also supported.) The device itself typically remains agnostic to the service or the fact that its resources or configurations are part of a service/concept at a higher layer.



Instantiated service models map to instantiated lower-layer network and device models. Examples include instances of paths, or instances of specific port configurations. The service model typically also models dependencies and layering of services over lower-layer networking resources that are used to provide services. This facilitates management by allowing to follow dependencies for troubleshooting activities, to perform impact analysis in which events in the network are assessed regarding their impact on services and customers. Services are typically orchestrated and provisioned top-to-bottom, which also facilitates keeping track of the assignment of network resources. Service models might also be associated with other data that does not concern the network but provides business context. This includes things such as customer data (such as billing information), service orders and service catalogues, tariffs, service contracts, and Service Level Agreements (SLAs) including contractual agreements regarding remediation actions.

Like intent, service models provide higher layers of abstraction. Service models are often also complemented with mappings that capture dependencies between service and device or network configurations. Unlike intent, service models do not allow to define a desired "outcome" that would be automatically maintained by the intent system. Instead, management of service models requires development of sophisticated algorithms and control logic by network providers or system integrators.

#### **4.2.2. Policy and Policy-Based Management**

Policy-based management (PBM) is a management paradigm that separates the rules that govern the behavior of a system from the functionality of the system. It promises to reduce maintenance costs of information and communication systems while improving flexibility and runtime adaptability. It is present today at the heart of a multitude of management architectures and paradigms including SLA-driven, Business-driven, autonomous, adaptive, and self-\* management [[Boutaba07](#)]. The interested reader is asked to refer to the rich set of existing literature which includes this and many other references. In the following, we will only provide a much-abridged and distilled overview.

At the heart of policy-based management is the concept of a policy. Multiple definitions of policy exist: "Policies are rules governing the choices in behavior of a system" [[Sloman94](#)]. "Policy is a set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects" [[Strassner03](#)]. Common to most definitions is the definition of a policy as a "rule". Typically, the definition of a rule consists of an event (whose occurrence triggers a rule), a set of conditions



(that get assessed and that must be true before any actions are actually "fired"), and finally a set of one or more actions that are carried out when the condition holds.

Policy-based management can be considered an imperative management paradigm: Policies specify precisely what needs to be done when and in which circumstance. Using policies, management can in effect be defined as a set of simple control loops. This makes policy-based management a suitable technology to implement autonomic behavior that can exhibit self-\* management properties including self-configuration, self-healing, self-optimization, and self-protection. In effect, policies define management as a set of simple control loops.

Policies typically involve a certain degree of abstraction in order to cope with heterogeneity of networking devices. Rather than having a device-specific policy that defines events, conditions, and actions in terms of device-specific commands, parameters, and data models, policy is defined at a higher-level of abstraction involving a canonical model of systems and devices to which the policy is to be applied. A policy agent on a controller or the device subsequently "renders" the policy, i.e., translates the canonical model into a device-specific representation. This concept allows to apply the same policy across a wide range of devices without needing to define multiple variants. In other words - policy definition is de-coupled from policy instantiation and policy enforcement. This enables operational scale and allows network operators and authors of policies to think in higher terms of abstraction than device specifics and be able to reuse the same, high level definition definition across different networking domains, WAN, DC or public cloud.

Policy-based management is typically "push-based": Policies are pushed onto devices where they are rendered and enforced. The push operations are conducted by a manager or controller, which is responsible for deploying policies across the network and monitor their proper operation. That said, other policy architectures are possible. For example, policy-based management can also include a pull-component in which the decision regarding which action to take is delegated to a so-called Policy Decision Point (PDP). This PDP can reside outside the managed device itself and has typically global visibility and context with which to make policy decisions. Whenever a network device observes an event that is associated with a policy, but lacks the full definition of the policy or the ability to reach a conclusion regarding the expected action, it reaches out to the PDP for a decision (reached, for example, by deciding on an action based on various conditions). Subsequently, the device carries out the decision as returned by the PDP - the device "enforces" the policy





and hence acts as a PEP (Policy Enforcement Point). Either way, PBM architectures typically involve a central component from which policies are deployed across the network, and/or policy decisions served.

Like Intent, policies provide a higher layer of abstraction. Policy systems are also able to capture dynamic aspects of the system under management through specification of rules that allow to define various triggers for certain courses of actions. Unlike intent, the definition of those rules (and courses of actions) still needs to be articulated by users. Since the intent is unknown, conflict resolution within or between policies requires interactions with a user or some kind of logic that resides outside of PBM.

#### **4.2.3. Distinguishing between Intent, Policy, and Service Models**

What Intent, Policy, and Service Models all have in common is the fact that they involve a higher-layer of abstraction of a network that does not involve device-specifics, that generally transcends individual devices, and that makes the network easier to manage for applications and human users compared to having to manage the network one device at a time. Beyond that, differences emerge. Service models have less in common with policy and intent than policy and intent do with each other.

Summarized differences:

- o A service model is a data model that is used to describe instances of services that are provided to customers. A service model has dependencies on lower level models (device and network models) when describing how the service is mapped onto underlying network and IT infrastructure. Instantiating a service model requires orchestration by a system; the logic for how to orchestrate/manage/provide the service model, and how to map it onto underlying resources, is not included as part of the model itself.
- o Policy is a set of rules, typically modeled around a variation of events/conditions/actions, used to express simple control loops that can be rendered by devices themselves, without requiring intervention by outside system. Policy lets users define what to do under what circumstances, but it does not specify a desired outcome.
- o Intent is a higher-level declarative policy that operates at the level of a network and services it provides, not individual devices. It is used to define outcomes and high-level operational goals, without the need to enumerate specific events, conditions,



and actions. Which algorithm or rules to apply can be automatically "learned/derived from intent" by the intent system. In the context of autonomic networking, ideally, intent is rendered by the network itself; also the dissemination of intent across the network and any required coordination between nodes is resolved by the network itself without the need for outside systems.

One analogy to capture the difference between policy and intent systems is that of Expert Systems and Learning Systems in the field of Artificial Intelligence. Expert Systems operate on knowledge bases with rules that are supplied by users. They are able to make automatic inferences based on those rules, but are not able to "learn" on their own. Learning Systems (popularized by deep learning and neural networks), on the other hand, are able to learn without depending on user programming. However, they do require a learning or training phase and explanations of actions that the system actually takes provide a different set of challenges.

## **5. Principles**

The following operating principles allow characterizing the intent-based/-driven/-defined nature of a system.

1. Single Source of Truth (SSoT) and Single Version/View of Truth (SVoT). The SSoT is an essential component of an intent-based system as it enables several important operations. The set of validated intent expressions is the system's SSoT. SSoT and the records of the operational states enable comparing the intended state and actual state of the system and determining drift between them. SSoT and the drift information provide the basis for corrective actions. If the intent-based is equipped with prediction capabilities or means, it can further develop strategies to anticipate, plan and pro-actively act on the diverging trends with the aim to minimize their impact. Beyond providing a means for consistent system operation, SSoT also allows for better traceability to validate if/how the initial intent and associated business goals have been properly met, to evaluate the impacts of changes in the intent parameters and impacts and effects of the events occurring in the system. Single Version (or View) of Truth derives from the SSoT and can be used to perform other operations such as query, poll or filter the measured and correlated information to create so-called "views". These views can serve the operators and/or the users of the intent-based system. To create intents as single sources of truth, the intent-based system must follow well-specified and well-documented processes and models. In other contexts



[[Lenrow15](#)], SSoT is also referred to as the invariance of the intent.

2. One touch but not one shot. In an ideal intent-based system, the user expresses its intents in one form or another and then the system takes over all subsequent operations (one touch). A zero-touch approach could also be imagined in case where the intent-based system has the capabilities or means to recognize intentions in any form of data. However, the zero- or one-touch approach should not be mistaken the fact that reaching the state of a well-formed and valid intent expression is not a one-shot process. On the contrary, the interfacing between the user and the intent-based system could be designed as an interactive and iterative process. Depending on the level of abstraction, the intent expressions will initially contain more or less implicit parts, and unprecise or unknown parameters and constraints. The role of the intent-based system is to parse, understand and refine the intent expression to reach a well-formed and valid intent expression that can be further used by the system for the fulfillment and assurance operations. An intent refinement process could use a combination of iterative steps involving the user to validate the proposed refined intent and to ask the user for clarifications in case some parameters or variables could not be deduced or learned by the means of the system itself. In addition, the Intent-Based System will need to moderate between conflicting intent, helping users to properly choose between intent alternatives that may have different ramifications.
3. Autonomy and Oversight. A desirable goal for an intent-based system is to offer a high degree of flexibility and freedom on both the user side and system side, e.g. by giving the user the ability to express intents using its own terms, by supporting different forms of expression of intents and being capable of refining the intent expressions to well-formed and exploitable expressions. The dual principle of autonomy and oversight allows to operate a system that will have the necessary levels of autonomy to conduct its tasks and operations without requiring intervention of the user and taking its own decisions (within its areas of concern and span of control) as how to perform and meet the user expectations in terms of performance and quality, while at the same time providing the proper level of oversight to satisfy the user requirements for reporting and escalation of relevant information. to be added: description for feedback, reporting, guarantee scope (check points, guard rails, dynamically provisioned, context rich, regular operation vs. exception/ abnormal, information zoom in-out, and link to SVoT. Accountable for decisions and efficiency, late binding (leave it to the



system where to place functionality, how to accomplish certain goals).

4. Learning. An intent-based system is a learning system. By contrast to imperative type of system, such as Event-Condition-Action policy rules, where the user define beforehand the expected behavior of the system to various event and conditions, in an intent-based system, the user only declare what the system should achieve and not how to achieve these goals. There is thus a transfer of reasoning/rationality from the human (domain knowledge) to the system. This transfer of cognitive capability implies also the availability in the intent-based system of capabilities or means for learning, reasoning and knowledge representation and management. The learning abilities of an intent-based systems can apply to different tasks such as optimization of the intent rendering or intent refinement processes. The fact that an intent-based system is a continuously evolving system creates the condition for continuous learning and optimization. Other cognitive capabilities such as planning can also be leveraged in an intent-based system to anticipate or forecast future system state and response to changes in intents or network conditions and thus elaboration of plans to accommodate the changes while preserving system stability and efficiency in a trade-off with cost and robustness of operations. Cope with unawareness of users (smart recommendations).
5. Explainability. Need expressive network capabilities, requirements and constraints to be able to compose/decompose intents, map user's expectation to system capabilities. capability exposure. not just automation of steps that need to be taken, but of bridging the semantic gap between "intent" and actionable levels of instructions Context: multi providers, need discovery and semantic descriptions Explainability: why is a network doing what it is doing
6. Abstraction - users do not need to be concerned with how intent is achieved

Additional principles will be described in future revision of this document addressing aspects such as: Target groups not individual devices, agnostic to implementation details, user-friendly, user vocabulary vs. language of the device/network, explainability, validation and troubleshooting, how to resolve and point out conflicts (between intents), reconcile the reality of what is possible with the fiction of what the user would want, "moderate", awareness of operating within system boundaries, outcome-driven





((what not how, for the user);(what and how/where, for the operator).not imperative/instruction based.).

The above principles will be further used to understand implications on the design of intent-based systems and their supporting architecture, and derive functional and operational requirements.

## **6. Lifecycle**

Intent is subject to a lifecycle: it comes into being, may undergo changes over the course of time, and may at some point be retracted. This lifecycle is closely tied to various interconnection functions that are associated with the intent concept.

Figure 1 depicts an intent lifecycle and its main functions. The functions are divided into two functional (horizontal) planes and into three (vertical) spaces.

The functional planes provide structure for the main functional concerns that are associated with intent: how to fulfill intent, and how to assure it.

- o Fulfillment is concerned with the functions that take intent from its origination by a user (generally, an administrator of the responsible organization) to its realization in the network. This includes:
  - \* Functions that recognize intent from interaction with the user and functions that allow users to refine their intent and articulate it in such ways so that it becomes actionable by an Intent-Based System. Those functions can involve unconventional human-machine interactions, in which a human will not simply give simple commands, but which may involve a human-machine dialog to provide clarifications, to explain ramifications and tradeoffs, and to facilitate refinements.
  - \* Functions that translate user intent into courses of actions and requests to take against the network, which will be meaningful to network configuration and provisioning systems. Possibly, this includes learning functions and algorithms that optimize the courses of actions to take in order to result in the best outcomes, specifically in cases where multiple ways of achieving those outcomes are conceivable.
  - \* Functions that perform and orchestrate the configuration and provisioning steps that were determined by the previous intent translation step.



- o Assurance is concerned with the functions that are necessary ensure that the network indeed complies with the desired intent once it has been fulfilled. This includes:
  - \* Functions that monitor and observe the network and its exhibited behavior.
  - \* Functions that assess and validate whether the observation indicate compliance with intent. This can include functions that perform analysis and aggregation of raw observation data.
  - \* Functions that trigger corrective action as needed.
  - \* Functions that abstract the observations and analysis results in a way that makes it possible for users to relate them to intent. In many cases, lower-level concepts such as detailed performance statistics and observations related to low-level settings need to be "up-leveled" to concepts the user can relate to and take action on.
  - \* Functions that report intent compliance status and that provide adequate summarization and visualization to the user.

The spaces indicate the different perspectives and interactions with different roles that are involved to address the functions:

- o The user space involves the functions that interface the network and intent-based system with the human user. It involves the functions that allow users to articulate and the intent-based system to recognize that intent. It also involves the functions that report back the status of the network relative to the intent and that allow users to assess whether their intent is having the desired effect.
- o The translation or Intent-Based System (IBS) space involves the functions that bridge the gap between intent users and network operations. This includes the functions used to translate an intent into a course of action, the algorithms used to plan and optimize those courses of actions also in consideration of feedback, the functions to analyze and abstract observations to validate compliance with intent and take corrective actions as necessary.
- o The Network Operations space, finally, involves the traditional orchestration, configuration, monitoring, and measurement functions which are used to effectuate the rendered intent and observe its effects on the network.



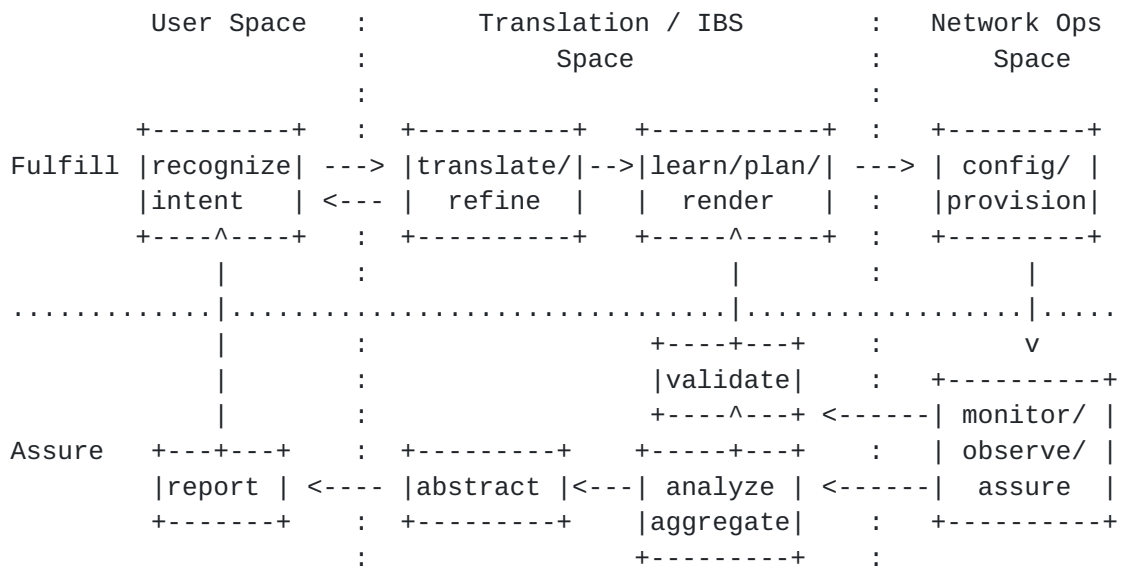


Figure 1: Intent Lifecycle

When inspecting the diagram carefully, it become apparent that the intent lifecycle in fact involves two cycles, or loops:

- o The "inner" intent control loop between IBS and Network Operations space is completely automated and does not involve any human in the loop. It involves automatic analysis and validation of intent based on observations from the network operations space, and feeding those observations into the function that plans the rendering of networking intent in order to make adjustments as needed in the configuration of the network.
- o The "outer" intent control loop involves also the user space and includes the user taking action and adjusting their intent based on feedback from the IBS.

Slight alternatives in intent lifecycles and the functions involved are conceivable. Figure 2 depicts one such alternative with an emphasis in intent fulfilment. (Todo: Intent attributes, intent states. Distinguish flow from users to network, and from network to user.)



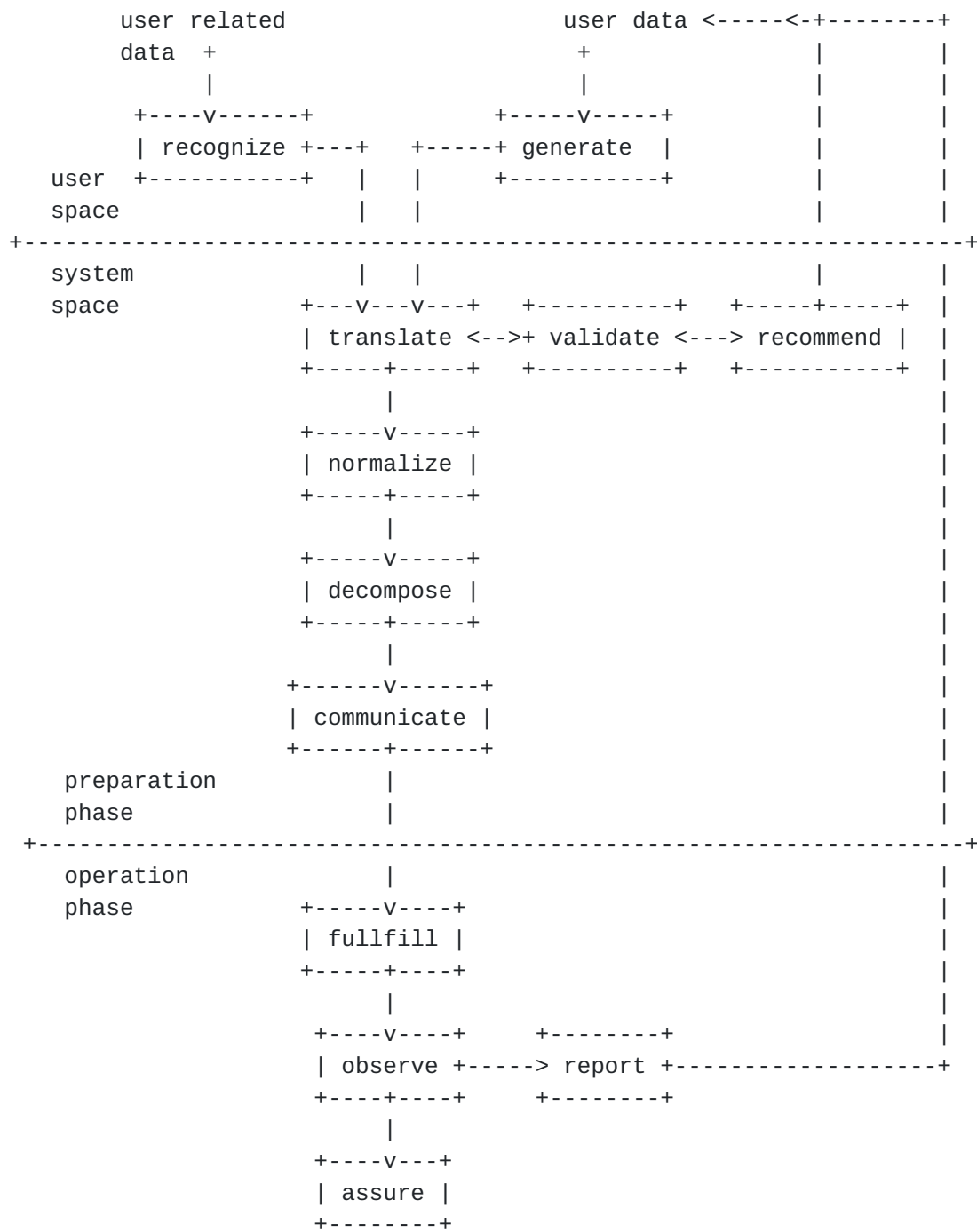


Figure 2: Intent Lifecycle (alt.)





## **7. Intent-Based Networking - Functionality**

Intent-Based Networking involves a wide variety of functions which can be roughly divided into two categories:

- o Intent Fulfillment provides functions and interfaces that allow users to communicate intent to the network, and that orchestrates the intent, i.e. that breaks down intent abstractions into lower-level network and device abstractions and performs or coordinates the configuration operations across the network.
- o Intent Assurance provides functions and interfaces that allow users to validate and monitor that the network is indeed adhering to and complying with intent. Control plane or lower-level management operations can cause behavior that inadvertently conflicts with intent which was orchestrated earlier. Accordingly, "intent drift" may occur. Network operators need to be able to detect when such drift occurs, or is about to occur, and be provided with the necessary functions to resolve such conflicts. This can occur by either bringing the network back into compliance, or by articulating modifications to the original intent to moderate between conflicting interests.

The following sections provide a more comprehensive overview of those functions.

### **7.1. Intent Fulfillment**

RBD

### **7.2. Intent Assurance**

Ability to reason about system' state by employing closed-loop validation in the presence of an inevitable change is a fundamental property of an Intent Assurance part of an IBN system. Since service expectations are created during intent consumption and modeling phase, closed-loop intent validation should start immediately, with the service instantiation. Telemetry consumed could then be enriched with an additional context and must always be processed in context of the Intent it has been instantiated. Direct relationship between the Intent and telemetry gathered enables correlation between changes in states and the Intent and provides contextual base for reasoning about the changes.



## **8. Items for Discussion**

Arguably, given the popularity of the term intent, its use could be broadened to encompass also known concepts ("intent-washing"). For example, it is conceivable to introduce intent-based terms for various concepts that, although already known, are related to the context of intent. Each of those terms could then designate an intent subcategory, for example:

- o Operational Intent: defines intent related to operational goals of an operator; corresponds to the original "intent" term.
- o Rule Intent: a synonym for policy rules regarding what to do when certain events occur.
- o Service intent: a synonym for customer service model [[RFC8309](#)].
- o Flow Intent: A synonym for a Service Level Objective for a given flow.

Whether to do so is an item for discussion by the Research Group.

## **9. IANA Considerations**

Not applicable

## **10. Security Considerations**

Not applicable

## **11. References**

### **11.1. Normative References**

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### **11.2. Informative References**



## [Boutaba07]

Boutaba, R. and I. Aib, "Policy-Based Management: A Historical perspective. Journal of Network and Systems Management (JNSM), Springer, Vol. 15 (4).", December 2007.

## [eTOM]

TMForum, "GB 921 Business Process Framework, Release 17.0.1.", February 2018.

## [I-D.ietf-teas-te-service-mapping-yang]

Lee, Y., Dhody, D., Fioccola, G., WU, Q., Ceccarelli, D., and J. Tantsura, "Traffic Engineering (TE) and Service Mapping Yang Model", [draft-ietf-teas-te-service-mapping-yang-02](#) (work in progress), September 2019.

## [Lenrow15]

Lenrow, D., "Intent As The Common Interface to Network Resources, Intent Based Network Summit 2015 ONF Boulder: IntentNBI", February 2015.

## [RFC7575]

Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", [RFC 7575](#), DOI 10.17487/RFC7575, June 2015, <<https://www.rfc-editor.org/info/rfc7575>>.

## [RFC8299]

Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", [RFC 8299](#), DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.

## [RFC8309]

Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", [RFC 8309](#), DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.

## [RFC8345]

Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", [RFC 8345](#), DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.

## [Sloman94]

Sloman, M., "Policy Driven Management for Distributed Systems. Journal of Network and Systems Management (JNSM), Springer, Vol. 2 (4).", December 1994.

## [Strassner03]

Strassner, J., "Policy-Based Network Management. Elsevier.", 2003.



[TR523] Foundation, O. N., "Intent NBI - Definition and Principles. ONF TR-523.", October 2016.

Authors' Addresses

Alexander Clemm  
Futurewei  
2330 Central Expressway  
Santa Clara, CA 95050  
USA

Email: ludwig@clemm.org

Laurent Ciavaglia  
Nokia  
Route de Villejust  
Nozay 91460  
FR

Email: laurent.ciavaglia@nokia.com

Lisandro Zambenedetti Granville  
Federal University of Rio Grande do Sul (UFRGS)  
Av. Bento Goncalves  
Porto Alegre 9500  
BR

Email: granville@inf.ufrgs.br

Jeff Tantsura  
Apstra, Inc.

Email: jefftant.ietf@gmail.com



