

NMOP
Internet-Draft
Intended status: Informational
Expires: 23 April 2026

N. R. Davis
Ciena
C. Cardona
NTT
D. Lopez
Telefonica
M. Palmero
Independent
20 October 2025

Generalized Capability Principles
draft-davis-nmop-generalized-capability-principles-00

Abstract

This document introduces a framework for capability modeling based on the specification and refinement principles established in ITU-T G.7711 Annex G (also published as ONF TR-512.7 see latest release) and the modeling boundaries work documented in draft-davis-netmod-modelling-boundaries. The framework defines how component capabilities can be explicitly described and refined via a process of pruning, refactoring, and occurrence formation.

These capability definitions can target detailed operational considerations, system interactions, licensing, abstract product declarations, or sales and marketing. The framework supports modular, layered, and fractal declarations of networked behavior, and provides a foundation for a suite of future IETF drafts aligned with ongoing work on photonic plug manifests, entitlement/licensing, IVY equipment modeling, energy/thermal considerations and related domains.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://github.com/marisolpalmero/draft-ietf-davis-generalized-capability-principles/blob/main/draft-davis-nmop-generalized-capability-principles-latest.md>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-davis-nmop-generalized-capability-principles/>.

Discussion of this document takes place on the Network Management Operations mailing list (<mailto:nmop@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/nmop/>. Subscribe at <https://www.ietf.org/mailman/listinfo/nmop/>.

Source for this draft and an issue tracker can be found at <https://github.com/marisolpalmero/draft-ietf-davis-generalized-capability-principles>.

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 23 April 2026.

Copyright Notice

Copyright (c) 2025 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Terminology	3
2. Introduction	4
3. Problem Statement	5
4. Specification in terms of the Model	6
5. Generalized Modeling via Component 寔鉄system寔鉄pecification Refinement	7
6. Some specification examples	9
7. Recursive narrowing	9
8. Specification of an assembly	10
9. Generalization of the specification	10
10. Characteristics of a language of specification	10

11. Specification language options	10
12. Building a specification structure	10
13. A specification evolution example	10
14. A system specification example	10
15. Broader Application of the Language	11
16. Conclusion	11
17. Security Considerations	11
18. IANA Considerations	11
19. References	11
19.1. Normative References	11
19.2. Informative References	11
Appendix A. Appendix A: Interpretive Notes on Refinement and Occurrence	12
A.1. A.1 No Single Refinement Path	12
A.2. A.2 Occurrence at Every Layer	13
A.3. A.3 Sweating Out the Shape	13
A.4. A.4 Classification Considered Harmful	13
Appendix B. Acknowledgments	13
Contributors	13
Authors' Addresses	13

1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in the document are to be interpreted as described in RFC2119}}.

The following terms abbreviations are used in this document:

- * capability: What can be achieved by an individual item both alone and in assembly (using the component-system pattern)
- * needs: Related to capability, this is what the item, either alone or in assembly, needs to achieve its capabilities
- * manifest: A list of essential contents
- * specification: A detailed description including arrangement
- * representation: An expression of properties from a perspective
- * occurrence: Placeholder
- * component-system: A pattern that expresses each item as a component where components can be assembled into systems and where a system can be represented as a component where that assembly may be of real things or may be abstractions of the effect of real things.

- * pruning:Placeholder
- * refactoring:Placeholder
- * pruning & refactoring: The process that supports progression from one view to the next view
- * capability/needs specification: A detailed description of what can be achieved by an individual item both alone and in assembly with respect to specific needs

2. Introduction

Currently, capabilities are mainly described loosely in human readable text, where that text is often incomplete, ambiguous or inconsistent. While people make these systems work in practice, the looseness result in errors, inefficiencies and limited reuse. As automation increases, there is a growing need to enable machine reasoning about the capabilities of network systems and components. While Large Language Models (LLMs) can interpret traditional documentation, there remains a strong need for greater formal rigor and structured representation to improve efficiency and precision. When asked, LLMs indicate that a rigorous model is preferable to loose ambiguous text. Existing IETF models predominantly focus on configuration, operational state, and telemetry. What is missing is a cohesive framework for expressing what a system can do, i.e., its capabilities, in a declarative, structured, and reusable form. This document introduces the principles for a capability modeling framework grounded in the specification concept established in [ITU-T_G.7711] ([ONF_TR-512]). It applies these principles through the lens of the **component system pattern** from [ONF_TR-512.A.2], using the concept of **emergence through recursive narrowing and occurrence formation**. These ideas are extended further by the modeling boundary principles described in [mobo]. The result is a standardized and extensible approach for expressing features, operational constraints, internal dependencies, etc. - separately from instance realizations. This approach supports capability modeling for any aspect of the controlled networking solution, and is designed to enable capability assembly, dynamic composition, licensing control, and integration with other IETF frameworks such as IVY equipment, photonic plug manifests, and entitlement interfaces. It also supports initiatives focussing on energy/thermal considerations where specific detailed capabilities and their power/thermal implications become critical considerations.

3. Problem Statement

Network technologies and management-control frameworks increasingly rely on declarative data models to represent both configuration and operational state. However, these models often lack a principled way to describe the capabilities of components and systems 寔背hat they are able to support or provide, independent of any particular operational instance. This omission makes it difficult to reason about compatibility, constraint satisfaction, composition, or even basic intent feasibility. Clearly, many of these activities take place prior to the installation of the equipment and indeed determine which equipments are to be planned to be installed. In these cases it is not possible to interrogate the actual equipment. Whilst knowing the YANG model for the equipment is beneficial, it is not sufficient as the YANG model essentially provides a space within which actual state etc. can be expresses, but it supports all possible combinations. The equipment will be very limited in comparison. Often it is desirable from a systems operation perspective to reduce the available capability through policy or other mechanisms due to the restrictions of a specific role. This becomes challenging if the base capability of a component is unclear and expressed in a chaotic form. In practice, five distinct concerns are often conflated, and also not fully expressed, within data models:

- The **generic definition** of a model element or concept (e.g., a termination point) - this is expressed in YANG. It is a very broad definition encompassing all possible opportunities and ofthen many illegal state combinations etc.
- The **capability definition** of a system or component, i.e., what it can support or expose (e.g., by a specific type or role of termination point). This is not expressed fully in YANG. There are both challenges with the expression of base capability and expression of the capability of combinations. This is especially sparse in representation - The users **policy definition** for system operation - the user may eliminate particular capabilities due to complexity, lack of trust, regulation etc. and will not want them offered or may not want them offered under certain circumstances. The equipment will be expected to behave as if it does not have the capabilities as appropriate.
- The **system combination** where an entity type may play several different roles and in each role may have specific distinct intentional limitations/restrictions.
- The **operational instance** 寔背hat is configured or active at a given time. Without a clear structural separation and with the sparseness of information on specific capabilities, it becomes challenging to formally describe feature constraints, support boundaries, or internal limitations. Implementers resort to informal documentation, code comments, yellow stickies, or out-of-band agreements to capture the intent behind model behavior. This reduces interoperability, increases integration effort, and undermines automation as a result of - **Ambiguity**

between what a model element *_is_* versus what a system *_can support_*.
- **Redundancy** and inconsistency in the representation of common constraints (e.g., port types, layering, resource limits). - **Tooling difficulty** when extracting interoperable subsets of large models or generating technology-specific profiles. - **Incompatibility** between modular subsystems or plug-ins that must declare and verify their supported features. Furthermore, current models tend to assume a fixed taxonomy of types and features, rather than supporting a process of recursive refinement. This limits their ability to express how complex capabilities *_emerge_* through constraint, composition, and modular pruning of more general-purpose constructs. What is needed is a modeling framework that: - Allows systems and components to be described in terms of their **capability boundaries**, including **capability interactions** separate from operational state, - Supports **refinement via pruning and refactoring to yield flexible structural transformation** rather than rigid inheritance or classification, - Enables **recursive occurrence formation**, where each stage of narrowing produces a usable semantic structure, - Accommodates **multiple valid refinement paths**, supporting different levels of granularity and domain specificity, - Provides a **coherent trace** from abstract capability declarations down to deployable or licensable configurations. This draft introduces such a framework by building on the refinement logic of [ITU-T_G.7711] ([ONF_TR-512]) in general and especially the **specification pattern** structures of ITU-T G.7711 Annex G (ONF TR-512.7) which provides a means of expressing bounded capability envelopes through a formal refinement of generic model elements. This also provides grounding in the recursive occurrence model informed by the component *囊都* system pattern [ITU-T_G.7711] ([ONF_TR-512.A.2] and modeling boundaries approach [mobol]. This document leverages the foundations laid by [ITU-T_G.7711] ([ONF_TR-512]).

The same expression challenges appear in statements of intent. The process of formulating intent through negotiation and resultant gradual refinement has a similar feel to the degrees of narrowing of the specification.

4. Specification in terms of the Model

The specification of capability should be presented in terms of the terminology of the problem space and hence in terms of the appropriate model. The challenge is determining which model is the "appropriate" model.

An area of the problem space can be described in different ways depending upon what the intention of the model is. There are many ways of representing a semantic space/

Prior to embarking on evaluation of specification of capability, it is important to consider the specific model and how it is structured.

- * Focus: Semantic area covered at centre and periphery
- * Specialization: Specific detailed focus on an area with rich structure, e.g., PCE, problem analysis, etc.
- * Granularity: the 寥徻ize寥 of the semantic units (including the depth of recursion of fractal representations)
- * Phase: The positioning of the semantic boundaries
- * Richness: The detailed coverage within a semantic unit
- * Fidelity: Precision v approximation
- * Abstraction: Closeness to actual detail
- * Maturity: Lifecycle development stage. How stable the model is likely to be. This is primarily about semantics, but also covers syntax.
- * Omission: Gaps and missing parts

5. Generalized Modeling via ComponentSystemSpecification Refinement

This framework moves away from rigid classification schemes and instead adopts a dynamic, refinement-based approach to modeling. Traditional classification attempts to impose fixed categories onto a system, but this often obscures nuance, variation, and the emergence of intermediate structures that carry operational or architectural significance.

We begin instead with the concept of a **universal component**—a general-purpose structure with maximal capability potential. Through the process of **pruning & refactoring** (constraint-driven refinement), this semantic volume is gradually narrowed, yielding intermediate structures with more sharply defined roles and properties. These refined artifacts are not pre-classified entities, but **emergent forms** that arise naturally at specific “sweat spots” in the refinement trajectory, where the remaining capabilities align with a recognizably useful or interoperable function.

Each such emergent form is treated as an **occurrence**. Occurrences appear at every stage of meaningful refinement including at the level of final implementation instances. At all stages of use the application of properties is via the idea of intent where even the

tightest constraint of a single value is essentially a statement of intent (as it is impossible to guarantee that a property will be set). This intent consideration will be dealt with further later in this document.

An LTP (Logical Termination Point) in [ITU-T_G.7711] ([ONF_TR-512]), for example, is not a primitive class but a pattern that arises from pruning and constraining the universal component until only the semantic envelope of an LTP remains. A TerminationPoint from RFC8345

To support variation, reusability, and convergence across implementations, each component or system is described not by a single fixed class, but by a **specification**: a constrained and possibly pruned refinement of a more general and broader model element. This allows the model to express bounded capabilities without requiring full instantiation, enabling tools and orchestrators to reason about compatibility, substitution, and support constraints before deployment. The specification describes the capabilities of an occurrence in terms of occurrences achieved via similar pruning. A system spec is a pattern assembly of subtly specialized occurrences at a particular level of specialization arranged in a meaningful structure that yields a relevant behaviour. The specification of an occurrence is itself a system spec.

The combination of the **componentsystem pattern** with the **specification refinement pattern** enables a modeling architecture where:

- * Systems are recursively composed of components,
- * Specifications constrain and refine capabilities at each level,
- * Occurrences are layered realizations of specs applied to specific contexts or configurations.

This approach supports **gradual realization**, where capability declarations can progressively transition from abstract to concrete, through intermediate spec refinements and pruning. Each layer of model realization adds specificity—structurally (via system composition), behaviorally (via constraints), and operationally (via mapping to configuration/state models).

A specification may provide explicit definifinition of a property as discussed above but it may also refer to one or more other specification(s). For example a specification may include a set of properties specified elsewhere. It may also define a property that is an enumeration of literals or identifies where those literal values or identify values are actually references to other specifications that provide deeper detail.

In an ideal environment, there is an ecosystem of specificactions each providing interrelated detail to fully define the semantics. The ecosystem would include specifications from standards bodies providing the definition of a network protocol that can be interpreted by an AI component such that the abstracted effect on the solution can be fully understood and simulated/emulated. Any detected conditions would be understood in terms of the protocol and hence the implications of the condition detected in terms of the carried signal can be fully understood.

Today's solution at best have a coded form of the semantic mantic interpretation that may not reflect the formal definition due to inaccuracies of interpretation. Many semantics are reduced to inconsistent labels that a user has to interpret. Whilst an LLM can do a reasonable job at interpretation of chaotic data, it will benefit a rigorous model traceable through formal definitions to fundamentals.

6. Some specification examples

This section will provide some examples and will reference the equipment capability draft and other future drafts.

7. Recursive narrowing

This builds on the example sketches and formalizes the process of recursive narrowing. Ahow the essential process. Use examples to illustrate the process -Thing to Component to Function to TP to specific TP to application of TP to instance of TP. -Thing to Component to physical thing to equipment to specific equipment type to use of that equipment to instance of equipment -A plug example Circle back and relate this more rigorous section to the specification examples.

8. Specification of an assembly

Build on the examples and the recursive narrowing to explain the subtle narrowings in a system/scheme spec. Describe the essential process. Use examples to illustrate the progression: - Same examples as recursive narrowing but focus on role and subtle specializations in role List other examples.

9. Generalization of the specification

Build a specification structure from the examples and show the references and reuses. Explain how the specification relates to the things in the problem space. Lay out the specification structure.

10. Characteristics of a language of specification

The language needs inherent capabilities (as opposed to after the fact bolt-on warts) Extract key characteristics from above and from mobo - narrowing requires specific redefine (relate to pruning) - occurrence is an assembly of constrained type and specific values - need to reference other specs as reusable parts - refactoring, minor specialization and assembly - interrelationship and influence - uncertainty and preferences (Need to review mobo and TR-547 spec, component-system etc.)

11. Specification language options

Landscape of languages... does anything do this? Take YANG and enhance (as discussed in mobo)

12. Building a specification structure

Tooling and support to build and interrelate. Catalogue/library of specs Deep application... machine interpretable structure in all standards Use of AI to reverse engineer specs with guidance... peer review and testing cycle

13. A specification evolution example

Discuss how a spec may change as understanding emerges and how it may be refactored.

14. A system specification example

Take the language considerations and set out system specs in a more formal way

15. Broader Application of the Language

Negotiation Refinement of planning Development of standards
Expression of uncertainty and pattern

16. Conclusion

Mindset Change Language challenges Use of AI Target is an ecosystem
of specs driving agentic components...

17. Security Considerations

TBD

18. IANA Considerations

This document has no IANA actions.

19. References

19.1. 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/rfc/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/rfc/rfc8174>>.

19.2. Informative References

- [BaseInventory] Yu, C., Belotti, S., Bouquier, J., Peruzzini, F., and P. Bedard, "A Base YANG Data Model for Network Inventory", Work in Progress, Internet-Draft, draft-ietf-ivy-network-inventory-yang-11, 14 October 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-ivy-network-inventory-yang-11>>.
- [ITU-T_G.7711] "Generic...", 31 August 2022, <<https://www.itu.int/rec/T-REC-G.7711/recommendation.asp?lang=en&parent=T-REC-G.7711-202202-I>>.
- [ivy] "ivy", 31 August 2022, <[https:// 3.pdf](https://3.pdf)>.

- [LF_TAPI] "Transport API", n.d., <<https://github.com/Open-Network-Models-and-Interfaces-ONMI/TAPI-Home>>.
- [mobo] "draft-davis-netmod-modelling-boundaries", 31 August 2022, <<https:// 3.pdf>>.
- [ONF_TR-512]
"TR-512 Core Information Model (CoreModel) v1.5", n.d., <https://opennetworking.org/wp-content/uploads/2021/11/TR-512_v1.5_OnfCoreIm-info.zip>.
- [ONF_TR-512.7]
"TR-512.7 Specification", n.d., <https://opennetworking.org/wp-content/uploads/2021/11/TR-512_v1.5_OnfCoreIm-info.zip>.
- [ONF_TR-512.8]
"TR-512.8 Control", n.d., <https://opennetworking.org/wp-content/uploads/2021/11/TR-512_v1.5_OnfCoreIm-info.zip>.
- [ONF_TR-512.A.2]
"TR-512.A.2 Appendix: Model Structure, Patterns and Architecture", n.d., <https://opennetworking.org/wp-content/uploads/2021/11/TR-512_v1.5_OnfCoreIm-info.zip>.
- [plug] "plug", 31 August 2022, <<https:// 3.pdf>>.

Appendix A. Appendix A: Interpretive Notes on Refinement and Occurrence

A.1. A.1 No Single Refinement Path

In this modeling approach, there is no single correct way to refine a universal component. The refinement process supports multiple valid paths, each representing a different semantic purpose, level of granularity, or domain context. What emerges depends not on a fixed taxonomy, but on the alignment of constraints, intent, and reuse patterns.

This enables: - Coexistence of multiple specification layers derived from the same abstract element, - Domain-specific "semantic phases" that are meaningful within a particular stack (e.g., optical vs packet), - Purpose-driven modeling: e.g., one path for plug manifests, another for logical topology.

A.2. A.2 Occurrence at Every Layer

Occurrences are not limited to final instances. Each meaningful stage of refinement produces an occurrence—an intent-aligned, constrained projection of the universal component. Even so-called “instances” are not full realizations, but expressed intent within a given operational context.

A.3. A.3 Sweating Out the Shape

Useful structural forms (e.g., an LTP) are not pre-classified primitives. They emerge from the pruning process when remaining capabilities reach a “sweat spot” of balance—enough constraints to be meaningful, but not so much as to be frozen. This allows the model to remain adaptive while still supporting mapping, reasoning, and automation.

A.4. A.4 Classification Considered Harmful

Rigid classification schemes tend to obscure natural emergence and lead to artificial separations. This model rejects top-down typing in favor of bottom-up capability surfacing, grounded in refinement logic. Semantic rigor replaces taxonomic rigidity.

Appendix B. Acknowledgments

This document has been made with consensus and contributions coming from multiple drafts with different visions. We would like to thank all the participants in the IETF meeting discussions.

Contributors

Nigel Davis
Ciena
Email: ndavis@ciena.com

Authors' Addresses

Nigel Robert Davis
Ciena
Email: ndavis@ciena.com

Camilo Cardona
NTT
Email: camilo@gin.ntt.net

Diego Lopez
Telefonica
Email: diego.r.lopez@telefonica.com

Marisol Palmero
Independent
Email: marisol.ietf@gmail.com