OPEN NETWORKING FOUNDATION

Common Information Model Overview

Version 1.2
September 20, 2016

ONF TR-513

ONF Technical Recommendation:
Common Information Model Overview

Disclaimer

Open Networking Foundation
2275 E. Bayshore Road, Suite 103, Palo Alto, CA 94303
www.opennetworking.org

1 Introduction 4

2 References 5

3 Abbreviations 5

4 Common Information Model 5

4.1 Core model 6
4.2 Technology/application specific sub-models 6

5 Interface Creation Process 7

5.1 Pruning and refactoring 7
5.2 Data Schema 8
5.3 Interface encoding 9

6 Common information model team 9

7 Main changes between releases 9

7.1 Summary of main changes between v1.1 and v1.2 9

List of Figures

Figure 1.1: ONF-CIM structure, related guidelines and interface creation process 4

Document History

Version	Date	Description of Change
1.0	March 13, 2015	Initial version
1.1	Nov. 24, 2015	Version 1.1
1.2	Sept. 20, 2016	Version 1.2

1 Introduction

This document describes the overall structure of the ONF Common Information Model (ONF-CIM), provides guidelines on the process used to develop it and guidelines on the derivation of interfaces from it. Other more detailed guidelines, as described below, are also provided. These guidelines are intended to capture the best practices and will be updated to reflect the experience that is gained as the ONF-CIM evolves.

The main changes between TR-513v1.1 and TR-513v1.2 are listed in section 7.

The purpose of the ONF-CIM environment is to provide a common repository for any information models that are developed in ONF. Figure 1 below provides an overview of the ONF-CIM and shows how purpose and protocol specific interfaces may be derived from it.

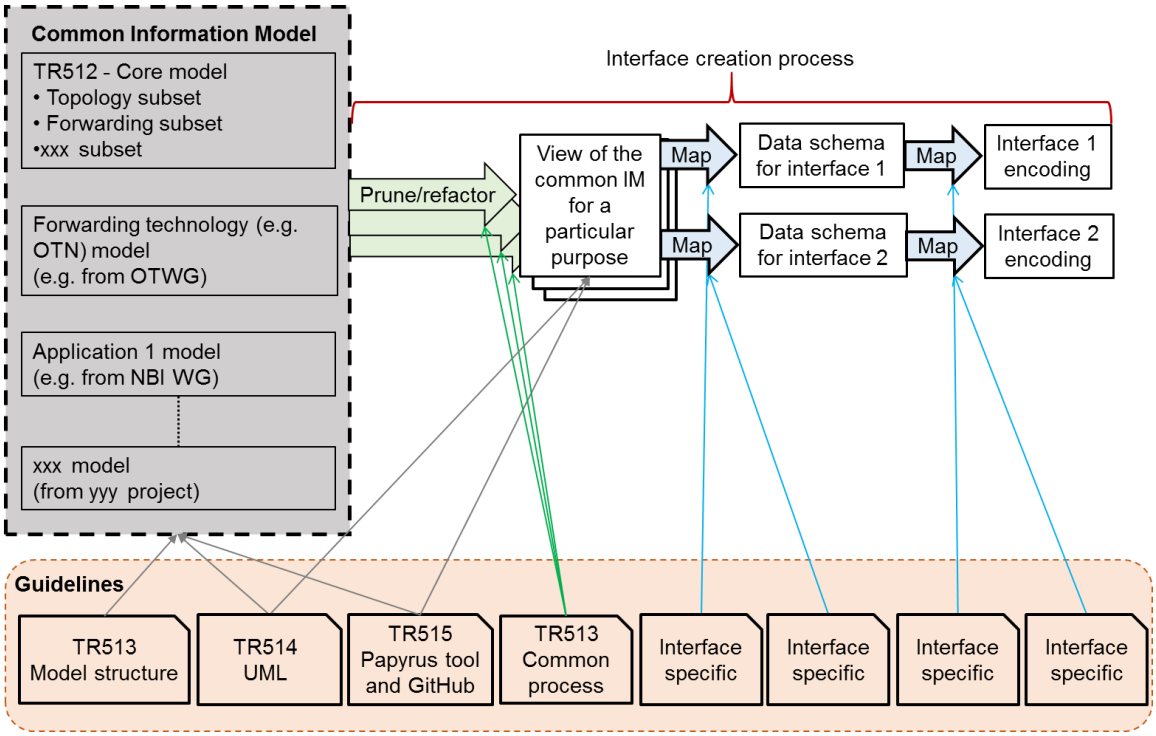

Figure 1.1: ONF-CIM structure, related guidelines and interface creation process

The first step in the interface creation process is to prune and refactor a copy of the ONF-CIM into a purpose specific IM view, then map this to a protocol specific data schema[1] and the encoding used for the interface. The process of mapping to a protocol specific data schema includes the mapping of the common operation pattern (defined in UML) to the operations supported by that protocol. The term Data Schema (DS) in this document is used in the context of either a specific protocol that is used to implement a purpose specific interface or, a programming language that is used to invoke a purpose specific API. The ONF-CIM (including the purpose specific views and protocol specific DS) are stored in an ONF-specific area of GitHub. Guidelines for the use of UML, the Papyrus tool and GitHub in the ONF-CIM are

[1] The term data schema is used instead of data model since the term data model is also used in a wider context and is sometimes used to refer to an information model

provided in [1] and [2]. High level guidelines for pruning and refactoring the ONF-CIM to provide a purpose specific view, and mapping to a data schema and mapping to a protocol or programming language, are provided in this document. Work to refine and automate both the pruning and refactoring process and the mapping to a protocol specific data schema is underway in ONF Open source projects and is available from https://community.opensourcesdn.org/wg/EAGLE/workgroup. Guidelines for mapping UML to YANG are provided in [3]

2 References

[1] ONF TR-514 UML Modeling Guidelines

[2] ONF TR-515 Papyrus Guidelines

[3] ONF TR-531 UML to YANG Mapping Guidelines

3 Abbreviations

API Application Programming Interface

ONF-CIM ONF Common Information Model

DS Data Schema

IM Information Model

IMP ONF Information Modeling Project

UML Unified Modeling Language

4 Common Information Model

An information model describes the things in a domain in terms of objects, their properties (represented as attributes), and their relationships. The ONF-CIM should include all of the artifacts (objects, attributes and relationships) that are necessary to describe the domain for the applications being developed.

It will be necessary to continually expand and refine the ONF-CIM over time, to add new applications, capabilities or technologies, or to refine it as new insights are gained.

To allow these extensions to be made in a seamless manner, the ONF-CIM is structured into a number of sub-model[2]. This modeling process allows the sub-models that contain these extensions to be developed, by the domain experts, with appropriate independence.

Over time, some parts of the ONF-CIM may need to be augmented or changed. Each IM team will ensure that any such areas are clearly identified using model lifecycle stereotypes (essentially controlled annotations). As the model matures any older artefacts will be marked as

[2] The Papyrus UML modeling tool supports the use of sub-models

deprecated to show that they shouldn't be used for new work, they will be maintained to ensure ongoing compatibility and to ease migration. The use of the lifecycle stereotypes is described in the UML modeling guidelines [1].

When a version of the ONF-CIM is released, all of the artifacts will be annotated with the model lifecycle stereotypes. All artifacts, independent of the lifecycle state (including, for example, those marked as experimental or deprecated), will be in the ONF-CIM release. Users of the ONF-CIM can use the lifecycle state to make an informed decision on which artifacts to use in an implementation.

4.1 Core model

It is expected that the artifacts in the core model will be used by multiple project teams. The core model is organized into a number of sub-models each addressing a specific topic to allow for easier navigation. The Information Modeling Project (IMP) will be responsible for maintaining the core model.

4.2 Technology/application specific sub-models

Technology/application specific projects will contribute to the ONF-CIM by developing sub-models which contain the artifacts (objects, attributes and relationships) that relate to their area of expertise.

It is expected that new sub-models will initially contain artifacts marked with the experimental stereotype. As the model is validated, for example by developing interfaces, the artifacts can be adjusted and moved through the lifecycle states. This allows for feedback between the model and an implementation. These technology/application specific sub-models will be placed in the common repository where they will be available for reuse by other ONF IM teams. The lifecycle stereotypes allow all users of the ONF-CIM to understand the lifecycle state of particular artifacts and make informed decisions on using them in implementations.

In some cases the addition of a technology/application sub-models will also require enhancement of the core model. The ONF-CIM IMP team will work with the technology/application team to ensure that these extensions are backwards compatible, can be reused by other groups and are not duplicated in multiple projects. These enhancements may need to be carried out by the ONF-CIM IMP team, supported by the IM teams developing the technology/application specific sub-models[3].

To assure coherency, any object, attributes or associations that might be identified during the development of technology/application specific views should be included in the appropriate sub-model of the ONF-CIM. Only those properties that relate to the specific encoding or style of interaction of an interface may be added outside the ONF-CIM.

[3] The core modeling team will often need to identify such cases and ensure that they work with the technology or application specific teams, as appropriate.

5 Interface Creation Process

A purpose specific information model is a true subset of the ONF-CIM and should be expressed in UML. These views should be developed by the relevant technology/application or interface specific project teams. Work to refine the guidelines and automate this process is underway in an ONF Open source project and is available from https://community.opensourcesdn.org/wg/EAGLE/workgroup.

A purpose specific information model will typically be much smaller than the entire ONF-CIM. If additional artifacts (objects, packages, attributes or associations) are identified while establishing a specific view, these artefacts should be added to the appropriate sub-model of the ONF-CIM. This will allow them to be used in that specific view and also make them available for reuse in other views as appropriate.

To provide maximum reuse, a purpose specific view should be developed in two steps:

a) Prune and refactor a copy of the artifacts of the ONF-CIM to provide a model of the network to be managed. Only those artifacts that represent the network capabilities that are both necessary to satisfy the purpose and are supported by that network are included in the purpose specific IM.
b) Define the access rights for the various groups of users that will manage that network

Pruning and refactoring provides a purpose specific IM that represents the capabilities of the network of interest.

The definition of access rights provides the ability to limit the actions that can be taken by the various user groups that will use that IM. For example a user group responsible for network configuration could be provided full read/write access and the ability to create or delete object instances; while a user group responsible for inventory may only be allowed read access (i.e. can see the network but cannot make changes).

5.1 Pruning and refactoring

The driver of pruning is to derive a (smaller) model with a narrower scope or view. Pruning can remove objects/packages/attributes/associations that are not required. Some high level guidelines for pruning are provided below:

- Select the required object classes from the common IM
  - All mandatory (non-optional) attributes and packages must be included
- Select the required conditional packages and optional attributes
  - Where appropriate, conditional packages and optional attributes in the common IM may be declared mandatory in the purpose specific IM
- Remove any optional associations that are not required
- Where appropriate, convert attributes defined as read/write in the ONF-CIM to read only in the purpose specific IM
- Narrowing capability, e.g., remove the right to create/delete some or all object instances
- Narrowing multiplicity. For example, from [0..*] to [1].
- Enlarging multiplicity is not allowed.

- The ONF-CIM should be updated if semantics are missing.

The refactoring process allows the model to be simplified and made compatible with existing models or terminology. For example, where the refactored view benefits from compressing the class model, or where there is some prior terminology and structure that is in force in the area of application and there is no opportunity to change that (at this point)[4]. Changes made during the refactoring process should be minimized. Some guidelines for refactoring:

- Collapsing of classes when reducing multiplicity (for example from [1..*] to [1])
  - When this results in a composition association of multiplicity [1] between a subordinate and superior object class, they can be combined into a single object class by moving the attributes of the superior class into the subordinate class
- Splitting of a class along a view boundary where the two parts are related by a specific multiplicity.
- Where beneficial, reducing the depth of the inheritance (i.e. combining object classes by moving the attributes of the superior object class into the subordinate object class)
- Adding reverse association navigability (if useful for the purpose)
  - In many places in the ONF-CIM, there is only support for navigation from a subordinate object class to a superior object class. This allows new subordinate object classes to be added without any impact on the superior object class. In a purpose specific implementation it is frequently useful to be able to navigate the relationship between superior and subordinate object classes in both directions.
- Constraining attribute definitions
  - Reducing legal value ranges
  - Defining which (if any) attributes should be read only (for all users)
  - Defining constraints between attributes
- Use the Realization association with a specific stereotype <<PruneAndRefactor>> to maintain the traceability from the pruned/refactored model to the ONF-CIM.

5.2 Data Schema

A Data Schema (DS) is developed in the context of either a specific protocol that is used to implement a purpose specific interface or, a programming language that is used to invoke a purpose specific API. It is possible to map directly from the purpose specific information model to an interface encoding. Work to refine these guidelines and automate the generation of a DS and interface encoding from the purpose specific information model is underway in an ONF Open source project and is available from https://community.opensourcesdn.org/wg/EAGLE/workgroup.

A DS should be developed by the relevant technology/application or interface project team. The DS is constructed by mapping the purpose specific information model together with the operations patterns from the ONF-CIM to provide the interface protocol specific DS that

[4] The long term vision is fully converged terminology across the industry thus removing the need of relabeling classes etc.

includes operations and notifications. The operations should include data structures taken directly from the purpose specific information model view with no further adjustment.

The development of the DS should consider the following:

- The operations should act on the information in a way consistent with the modeled object lifecycle interdependency rules as defined in the ONF-CIM.
  - Instance lifecycle dependencies should ensure sensible interface operation structuring and interface flow rules
  - Some form of Transaction should be used over the interface to account for instance lifecycle dependencies of the model
- The operations should abide by the attribute properties
  - Read only attributes (except those which are defined as "isInvariant") should not be included in data related to creation of an object (e.g. not in createData) or in a specification of a desired object structure outcome.
- Use of attribute value ranges, etc. to allow "effort" statement, optionality and negotiation to be supported by the interface

5.3 Interface encoding

This step encodes either a purpose specific data schema or, a purpose specific information model into either a specific protocol that is used to implement a purpose specific interface or, a programming language that is used to invoke a purpose specific API. If the interface is encoded directly from the purpose specific information model then the interface operations must be added as described above. This encoding should be developed by the relevant technology/application or interface specific project team.

6 Common information model team

The ONF-CIM will be maintained by the Information Modeling Project (IMP) team. It is expected that each of the other ONF project teams that will use (and contribute to) the ONF-CIM will provide a representative to ensure that the work of the IM sub-team meets the needs of their project.

The IMP IM sub-team will develop and maintain the core model and maintain the library of all of the model in GitHub. They will provide guidance (when requested) to the other IM sub-teams that are generating the technology/application specific sub-models of the common IM or developing a purpose specific view of the common IM.

7 Main changes between releases

7.1 Summary of main changes between v1.1 and v1.2

- Replace fragment with sub-model
- Reference the UML-YANG mapping guidelines
- Reference the work in the ONF Open source Eagle project

- Editorial changes to improve readability and consistent use of terms

