SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

The Power of YANG Configuration Templates

A Technical Paper prepared for SCTE by

Pawel Sowinski
VP Technical Excellence
Falcon V Systems
Sudbury, Mass
p.sowinski@falconvsystems.com

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

1. Introduction

Cable Operators pursuing Flexible MAC Architecture (FMA) deployments are facing the prospect of a N-fold increase in the number of managed devices in their HFC access networks. The increase is a result of a necessary replacement of integrated CCAPs with Distributed Access Architecture (DAA) components including MAC Managers and Remote MAC Devices (RMDs), where RMDs serve a much smaller service-delivery footprint. Typically, RMDs provide services for just one service group. As part of the DAA transition, N-number of RMDs displace each I-CCAP, a large-scale platform supporting N service groups. The number "N" could reach or exceed 128. This dramatic increase in scale requires a shift towards zero-touch, automated provisioning processes. While FMA specifications already provide tangible solutions for this problem at the interface between the MAC-NE and the MAC Manger, the paper examines the impact of this transition on the provisioning tasks within the operators' back-office.

Two additional key observations lay the technical foundation for this paper. First, a complete configuration dataset of each RMD is unique in that it includes many configuration attributes that are specific to a particular device and its deployment topology. The examples of such device-specific attributes include power levels configured on the RF components, the geographical location, and the inventory tag. Second, most of the device configuration attributes, including service group configurations, are common across large groups of RMDs.

The paper outlines a framework that capitalizes on abstraction of repeatable patterns from the standardized YANG configuration models of RMDs with creation of configuration templates corresponding to the identified patterns. YANG-based configuration templates coupled with unique device attribute values maintained in a relational database allow automated systems to generate complete device configuration datasets enabling effective management of the configuration lifecycle, including software or service upgrades.

The framework provides benefits to Cable Operators by reducing the overall size of fundamental configuration data and by providing structure better suitable for automation. For example, a Cable Operator deploying thousands of RMDs may be able to reduce the footprint of their fundamental configuration dataset to just a few dozen YANG-based configuration templates and use predictable, automated system to dynamically generate full-formed, operational configuration datasets from these templates. The outcomes are in the form of modular, automated, and streamlined provisioning system with increased agility and reduced OPEX.

The paper outlines a YANG-based configuration template framework, based on FMA YANG models developed by CableLabs®, while providing several real-world examples of the benefits.

2. Flexible MAC Architecture

CableLabs has released a suite of Flexible MAC Architecture specifications and continues to expand the specifications through industry collaboration. The FMA project is a key element of a larger CableLabs Distributed Access Architectures (DAA) program. The DAA program includes additional projects, such as Remote PHY (RPHY) and Coherent Optics as well as an industry-wide YANG model standards library and YANG model development pipeline.

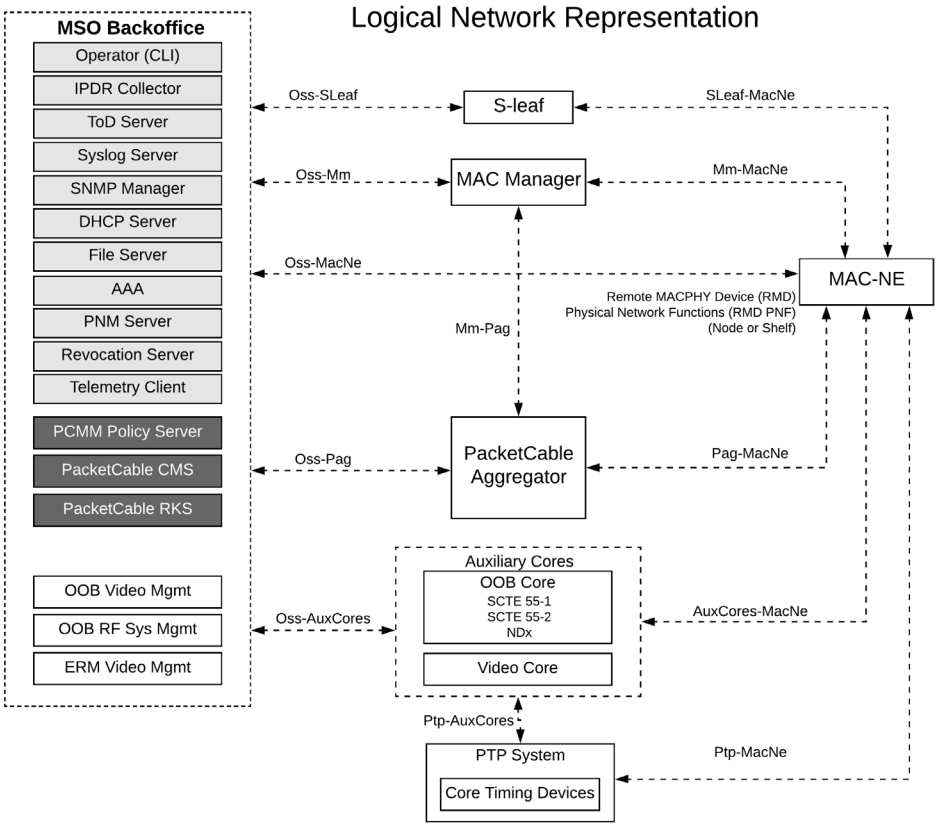

Figure 1 - FMA Phase 1 Reference Architecture

The suite of FMA specifications have achieved a level of stability with the Phase 1 scope officially concluding at the end of CY 2021. The Phase 1 FMA Reference Architecture, as defined in [FMA-SYS], is illustrated on Figure 1. A detailed description for each FMA system component and interface displayed in the Reference Architecture can be found in [FMA-SYS] and [FMA-OSSI].

The following sections of the paper will narrow the focus to those FMA components and interfaces (realized as APIs) which are relevant to the YANG-based configuration template framework. The goal is to align the framework against the FMA Reference Architecture.

Figure 1 depicts the interface between the MAC Manager and the MAC-NE (e.g., RMD). This interface is generally referred to as the MAC Manager to MAC-NE Interface (MMI), labelled as 'Mm-MacNe' in the diagram and specified in [FMA-MMI]. The interface's implementation relies on the RESTCONF protocol, and a library of MAC-NE YANG models developed by CableLabs. The IETF defines the RESTCONF protocol as specified in [RFC 8040].

The MAC Manager fulfills the role of a RESTCONF Client at the northbound of the Mm-MacNe interface, while the MAC-NE operates as the RESTCONF Server at the southbound. Among other primitives, the MAC Manager is responsible for transferring all configuration information to the MAC-NE over the MMI/Mm-MacNe interface. Since the RESTCONF protocol facilitates unicast communication, the MAC-NE YANG models have been designed for transfer of configuration to an individual MAC-NE (RMD).

Another FMA Reference Architecture interface relevant to the subject of the paper, extends between the MAC Manager and the operator's back-office systems, labeled as 'Oss-Mm' in Figure 1. In addition, Figure 1 lists several back-office applications used in the management of an FMA deployment, however it should be pointed out that a network Configuration Management System (CMS) is not included. The CableLabs' FMA specifications do not explain whether configuration management was purposefully omitted as out-of-scope for Phase 1, or under the assumption that configuration of MAC-NEs is an internal domain of the MAC Manager. This paper makes an assumption that a CMS is a mandatory part of the Cable Operators Network Management System (NMS) infrastructure and focuses on selected issues revolving around configuration management functions in FMA deployments.

Figure 2 illustrates an updated reference architecture diagram and outlines the scope of functionality covered by the paper.

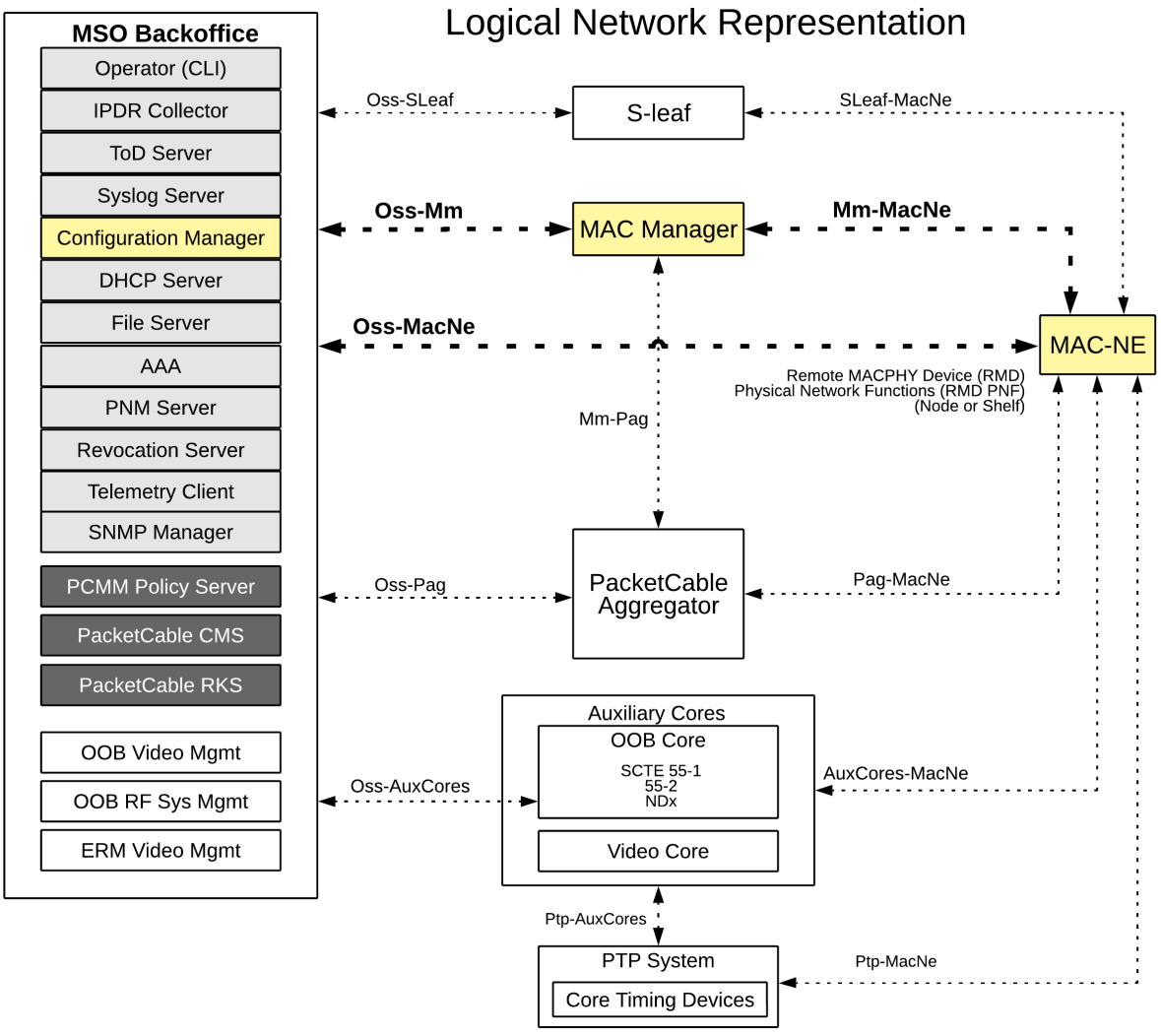

Figure 2 - YANG-based Configuration Template Framework Scope

3. Addressing the Problem of Scaling Configuration Management Data

Let us first briefly illustrate the scale of the provisioning problem faced by Cable Operators deploying FMA through examination of the following example. An HFC network of a medium-sized Cable Operator with a few million paying subscribers may consist of 100,000 service groups. If each service group corresponds to exactly one RMD and considering that an average size of an initial configuration dataset can undoubtedly exceed 500 KB (or 10,000 discrete configuration attributes), and the cumulative size of the initial configuration dataset for such an operator could amount to 500 KB * 100,000 = **50 GB** (or 1 billion discrete configuration attributes). This is the volume of the initial configuration information, often referred to as "day-one config", for all RMDs in this example. This data is encoded and transferred across the 'Oss-Mm' interface presented on Figure 2 via the RESTCONF protocol, in a unicast fashion, one-by-one between the FMA MAC Manager and each RMD in the Cable Operator's network whenever an RMD is initialized or reinitialized.

A data footprint of such scale can very well fit into the storage of an average smartphone. How is that a problem, one might ask?

While storage and transfers of several gigabytes of data cannot constitute a problem for any modern networked computing system, managing "the source of truth" of such sizable footprint of configuration data through the lifecycle of 100,000 independent systems is an entirely different ballgame.

The data abstraction approach presented in the paper takes advantage of the repeatable patterns in the deployment environment, the cookie-cutter like HFC access network design and the standardization of device configuration management interfaces using YANG data models. This methodology relies on breaking down the devices' configuration datasets into smaller, more manageable modules, extracting individual configuration attributes and their values into a standalone database and creating reusable configuration templates from common, shared datasets. The key enabler of this approach is YANG and its powerful, modular definition of data models.

3.1. The Intended Audience

The intended audience for this paper is the personnel of Cable Operators who are designing or preparing to deploy FMA based systems in their HFC networks, including network provisioning architects, engineers, application developers and business analysts. This paper assumes that readers have some working knowledge of YANG modeling and protocols designed to transport YANG data such as RESTCONF or NETCONF.

4. YANG-based Configuration Templates

4.1. What are Configuration Templates?

This paper adopts the term "Configuration Template" to be commonly understood as **a predefined prototype for instantiated configuration data** that can be further customized and applied to multiple systems in various scenarios.

Configuration Templates are broadly deployed across the software technology industry and distinctly in the networking and cable sectors. For example, some CCAP manufactures provide templates as means to reduce the size of the CLI configuration payloads for network interfaces and selected functions.

4.2. YANG-based Configuration Templates

FMA relies on YANG as a modeling language for MAC-NE status and configuration datastores. YANG was originally defined in [RFC 6020] as a modeling language for the description of data carried in the NETCONF protocol. Since then, YANG has graduated to version 1.1 [RFC 7950] and a wide array of RFCs and other technical standards have been developed to cover adjacent aspects such as encoding, metadata, versioning, translation and handling operational issue with multiple datastores. Today, the networking and cable industry operates with the collateral of thousands of YANG modules, some of which have been published or remain in private distributions.

However, there is no standard or formal definition for YANG-based Configuration Templates (YCTs). No IETF RFC, nor an open-source project or framework provides a formalized definition for a YCT. A YANG-based Configuration Template is a term coined herein. In this context, a YANG-based Configuration Template is a Configuration Template in which the instantiated configuration data is compliant to a particular YANG model and the data encoding generally follows the common encoding rules for YANG data as defined in relevant IETF standards. A YCT can be programmatically validated against the corresponding YANG model.

Each YCT consists of two sections:

1. Data Payload
2. Metadata

The YCT sections are explained in greater detail in the following sections of the paper.

4.2.1. Data Payload

The most essential part of a YCT is the data payload. The data payload contains the instantiated configuration dataset corresponding to a branch of the device's YANG schema. The data payload must not contain any status or operational state data, but partial datasets are permissible. The data payload of one template is typically compliant to a model defined by multiple YANG modules.

The data payloads can be represented in several human and machine-readable formats, such as XML, YAML or JSON. This paper uses JSON for the format of the payload data in the following examples due to its compactness and human readability. The template data in the paper adheres to the standard JSON encoding rules for YANG-defined data per [RFC 7951].

In the simplest form of a YCT, the data payload contains the complete configuration dataset for a branch of the device's YANG schema tree. The example in Figure 3 shows the data payload for configuration of MAC-NE security parameters. Such a template can be incorporated into the target configuration without further changes. The configuration target is defined in section 5.2.

```
"security": {
  "sav": {
    "sav-control": {
      "cm-auth-enable": true
    }
  },
  "tftp-security-config": {
```

```
    "config-file-learning-enabled": true
  },
  "cmts-encrypt": {
    "encrypt-alg-priority": "AES128_CBC_MODE DES56_CBC_MODE DES40_CBC_MODE"
  },
  "certificates": {
    "cert-revocation-method": "NONE",
    "online-cert-status-protocol": {
      "signature-bypass": false
    }
  }
}
```

Figure 3 – Yang-Based Configuration Template

More interesting scenarios ensue when YCT data payloads include variables and/or expressions.

A template variable constitutes a data node value, which is kept within the template as a human and machine-readable name in the form of a character string. Variable names in the paper examples, by convention, are presented as strings that start with the '$' character (e.g., $my-variable-name).

The values for the template variables are maintained outside of the template definition, in a Target Database, which is described in a later section of the paper. A variable designates a node value specific to the template target, which is typically an individual MAC-NE device or a group of MAC-NE devices. The Configuration Management System replaces variables with actual values during the process of rendering a template. The template rendering process and the template-oriented configuration management system are described in a later section of the paper.

The example provided in Figure 4 shows a data payload for a configuration of a MAC-NE for Precise Time Protocol (PTP) with a variable for the value of the attribute "ptp-master-addr". The name of the variable in this example is '$ptp-master-ip-address'.

```
"rdti-cfg": {
  "core-ptp-clk-cfg": {
    "ptp-clk-profile-id": "00:19:a7:02:01:00",
    "ptp-master-addr": "$ptp-master-ip-address",
    "ptp-master-priority": 0,
    "ptp-clk-priority1": 128,
    "ptp-clk-priority2": 255,
    "ptp-clk-domain": 24
  }
}
```

Figure 4 – YANG-Based Configuration Template Data Payload

When such a template is rendered for a selected target, the system retrieves the actual IP address of the PTP master clock from the Target Database entry and inserts it into the configuration dataset. With this approach, the PTP master clock server's IP address can be individually applied for each target and combined with common data from the template to form a complete configuration dataset for the device.

A template expression is a logical or a mathematical formula for algorithmic determination of a value of a data node in the template. An expression can include operations on constants, template variables and values of other data nodes contained within the template, including the key attributes of YANG lists. Similarly, as with variable replacement, expressions are evaluated during the rendering of a template for a selected target.

A familiar example of a template expression could be the formula to calculate the value of frequency of a downstream SC-QAM channel when several such channels are grouped into a single, continuous block. Such a formula can be the sum of the value of a variable representing the center frequency of the lowest channel in a block and channel index multiplied by 6 MHz.

$channel-frequency = $start-frequency + channel-index 6,000,000*

The syntax of expressions and their usage rules are quite complex. For this reason, and because expressions' encodings are not essential to the purpose of the paper, the detailed definition of template expressions is left out of the scope of the paper.

4.2.2. Metadata

Template metadata consists of a set of attributes that hold miscellaneous properties of a template outside of the configuration data payload. The most important metadata attributes are the **template-name** and the **template-schema-root**.

The template name serves as the primary identifier of the template within the system. The system refers to templates by their names. For this reason, each template name needs to be unique within the system.

The YANG modelled data maintains a tree-like hierarchy where each node can be uniquely identified with a schema path. The template schema root defines the parent node of the target's YANG data schema tree from which the data within the template is in scope. Thus, the template schema root is a YANG schema path. It can include keys and variables so that different templates can be rendered onto specific elements of a single YANG list. A template metadata defines exactly one or zero template schema roots. When the template is rendered for a selected target, the system can override the template schema root with a custom template schema root configured for the target. For example, the template schema root can refer to an individual entry of a list, or to all entries of a list. In the former case a template provides data for the referenced list entry. In both cases, the set of data nodes corresponding to the list can be formed from multiple templates.

Other metadata attributes provide information which can be helpful with administrative tasks. The example of template metadata shown below comprises several other attributes, such as **template-description**, **template-notes** and **template-revision-history**.

The metadata encoding is consistent with the JSON encoding rules for YANG data nodes. All metadata attributes, just like the data payload, are encoded as JSON key-value pairs. Template metadata encoding follows the rules from [RFC 7952].

An example of template metadata is shown in Figure 5.

```
"template-metadata": {
  "template-name": "ptp-config-101",
  "template-schema-root-point": "/mac-ne/networking/",
  "template-description": "This template defines configuration for MAC-NE PTP
operation for G.8275.2 PTP profile",
  "template-notes": "This template is incomplete. It is missing a number of
mandatory configuration attributes",
  "template-revision-history": [
    {
      "revision": "2022-01-01",
      "description": "Added ptp-clk-domain attribute.",
      "Author": "George D."
    },
    {
      "revision": "2021-10-01",
      "description": "Initial revision",
      "Author": "Bob D."
    }
  ]
}
```

Figure 5 – YANG-Based Configuration Template Metadata Payload

5. Template-Oriented Configuration Management System

This section describes an environment in which all data configuration originates from a template. The difference between a template and configuration payload is the name.

This section provides a rough framework for a system which creates MAC-NE configuration payloads from YANG configuration templates. The paper limits the description of the system to a minimum necessary to illustrate the principles of operation of a system based on Configuration Templates. In the context of this paper, the system is referred to as a Template-Oriented Configuration Management System, or TOC System. The TOC can become an integral part of the Cable Operator's Data Operations platform architecture.

The TOC System is presented on Figure 6 and consists of three main components:

1. Repository
2. Target Database
3. Template Management Application

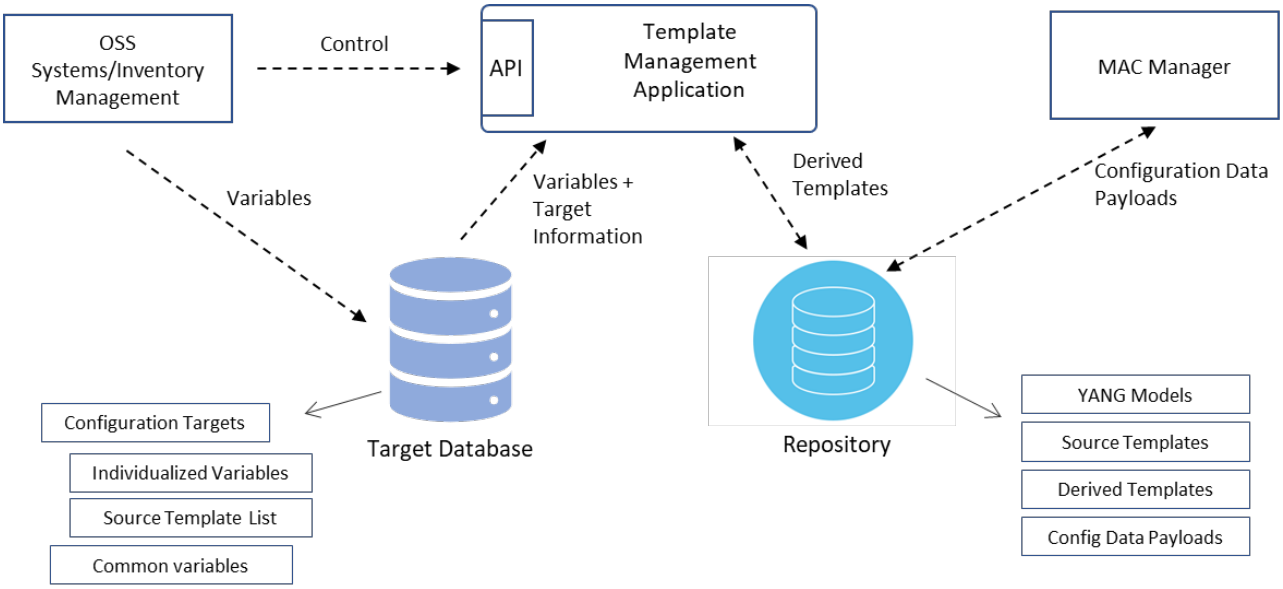

Figure 6 – Template-Oriented Configuration Management System

5.1. Repository

The Repository serves as a storage location for files which house YANG models, source and derived configuration templates, as well as the fully instantiated configuration payloads. The Repository can be implemented on top of a distributed Version Control System (VCS) such as Git or on top of a versioned object storage system such as Amazon S3. However, a distributed VCS provides the features required for a repository of templates. Such features incorporate sophisticated versioning tracking, management and security controls, including user access controls and what is most important, built-in integration for automated CI/CD pipelines. Therefore, the paper considers a VCS such as GitLab or GitHub as a default choice for the Repository.

Fully instantiated configuration payloads from the Repository are communicated to the FMA MAC Manager through a standardized API, e.g., RESTCONF.

5.2. Target Database

The Target Database contains information necessary for rendering configuration targets. The Target Database can be implemented as a relational database, such as an open-sourced Redis database.

This section defines the types of supported configuration targets. The TOC System operates on two categories of configuration targets: Derived Configuration Templates and Configuration Data Payloads.

Derived Configuration Templates are templates which the TOC System generates as the product of rendering of one or more source configuration templates. For example, a derived template representing the configuration of a downstream RF Port of a MAC-NE can be created from multiple source configuration templates where each source template contains configuration payload for one type of channel, for example, a list of downstream SC-QAM and OFDM channels and forward SCTE-55-1 OOB channels.

Another supported type of configuration target, the device Configuration Data Payload represents a portion of, or the entire configuration data instance intended for the target device. The Configuration Data Payload format is essentially the same as the Derived Configuration Template except all variables have been replaced with values, and all expressions have been fully evaluated. Configuration Data Payload must strictly conform to the branch of the YANG schema it represents and its intended use case.

The information maintained in the Target Database includes the following attributes:

- The target identification.
  - The name of the configuration target.
  - The location of the target file within the Repository.
  - Necessary metadata for creation of target payload, including YANG model information.
  - Target revision information.
- A list of source templates from which the target needs to render.
  - A substitution schema root that are optional for each source template. As mentioned earlier, each source template maintains the template-schema-root attribute, which can be overridden by the substitution schema root from the target.
- The YANG schema root for the target, in the form of the YANG path to the root node in the target.
- A list of variable replacement values for the target. When variable values are specific to exactly one target, then the database record of the target contains the values. In other cases, when variable values are shared by multiple targets, the Target Database record of the target provides an indirect reference to the value maintained in a common, shared record. This way the Target Database contains only one record with definition of the value, the single source of truth for multiple targets. The set of variable replacement values can grow quite large, when variables are leaves of multi-dimensional configuration lists.

Most of the information in the Target Database originates from the Cable Operator's BSS/OSS systems, while the information about the relationship between source templates and targets comes from human-assisted YANG schema breakdown into template hierarchy.

5.3. Template Management Application

The third component of the TOC System is the Template Management Application (TMA).

TMA implements all necessary configuration manipulation logic, including the rendering and the validation of target Configuration Data Payloads. The TMA also incorporates machine-to-machine APIs to the BSS/OSS systems through which the processes of rendering and validation of configuration payloads can be automatically initiated, and the status of these processes can be communicated back to the requestor.

5.4. Template Rendering

Template rendering is a procedure for creating target configuration payloads in accordance with their YANG models. It can be summarized as a five-step process:

1. The TMA retrieves the target information from the Target Database.
2. The TMA extracts the payload from all source template payloads identified in step 1.
3. The source payloads are combined in compliance with the applicable YANG model to create the target data payload.

4. The TMA injects the individualized target data into the payload, replacing variables with corresponding values from the Target Database.
5. The data payload of the derived template is written into the Repository.

The product of the rendering procedure is the target payload, which can be in the form of a derived template or a fully instantiated Configuration Data Payload. Next, the TMA validates the product, the target configuration payload against the YANG model and custom validation rules.

The combined source payloads typically form different branches of YANG instance data tree in the target's payload. However, the rendering process can also combine partial datasets from source template payloads that are part of the same branch.

A derived template can inherit selected variables from its source templates. Other variables can be replaced with actual values from the Target Database record associated with the target template. The TOC System decides whether a variable is replaced or inherited by examining the content of the Target Database record for the target. When the Target Database record contains a value for the variable, the variable is exchanged for the value. Otherwise, the variable is inherited by the target in the same form as in the source template.

Similarly, selected expressions can be evaluated during rendering of the source templates while other expressions are inherited by the Derived Template in the same form as present in the source template.

The rendering process can operate in multiple iterations. A Derived Configuration Template created because of rendering, can later become a source template in another iteration of the rendering process. The iterative process is particularly useful in creation of multi-level hierarchies of configuration data from simpler, more manageable, and "flat" templates. For example, the current FMA MAC-NE YANG model places many configuration attributes as high as at the fourth or even the fifth level of the YANG schema tree.

In the example shown on Figure 7, two source templates, named T1 and T2 are rendered into a derived template, named T3. The data payloads of T1 and T2 are combined in compliance with the YANG model. Next, Variable-1 of source template T1 and Variable-2 of source template T2 are replaced with Value-7 and Value-6 respectively. Variable-3 is inherited by derived template T3 from source template T2.

Note, that the payload from source template T1 is inserted at the same level as data in Container Z from source template T2 based on the T1's schema root.

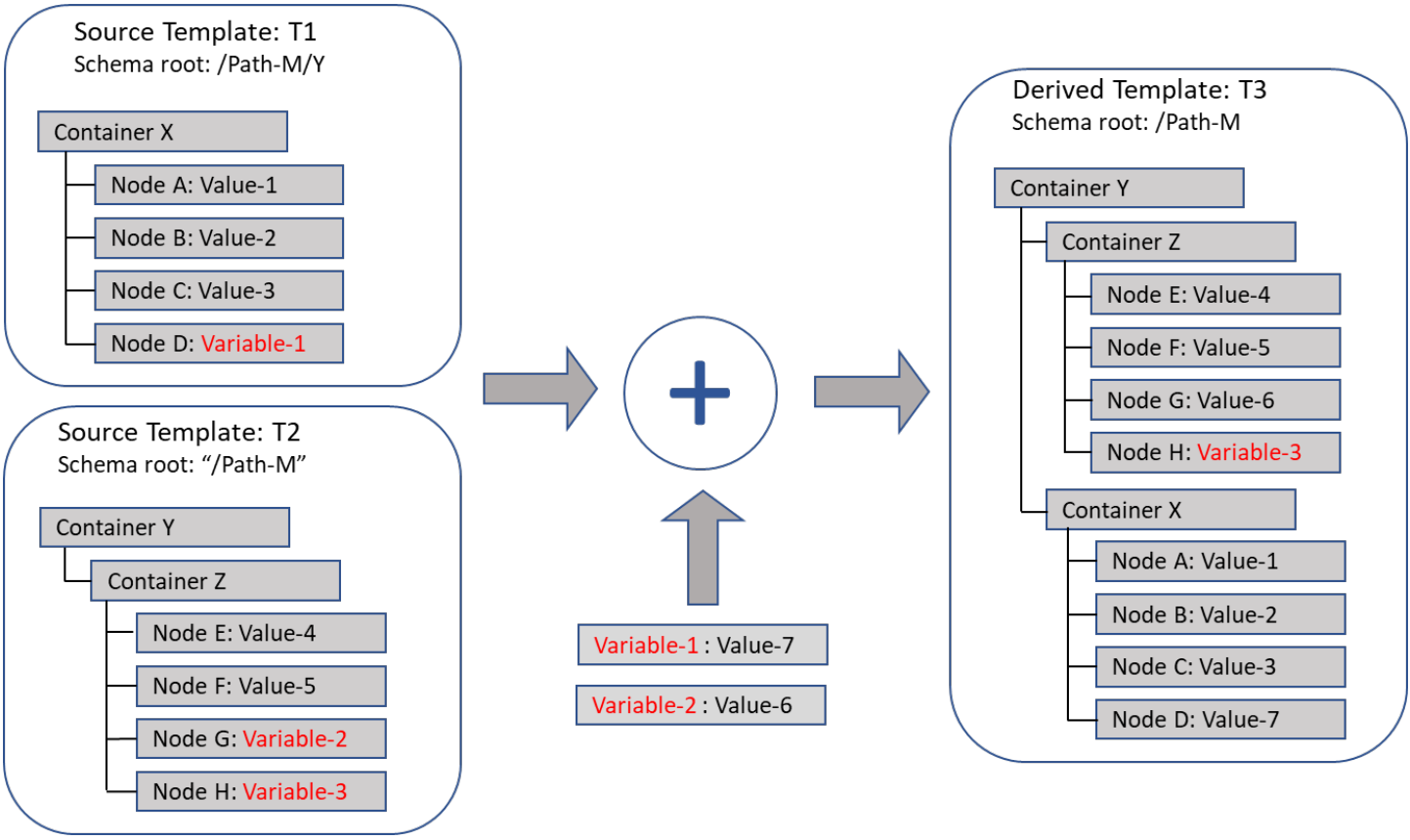

Figure 7 – Template Rendering Example

The instantiated data held in the template payloads, as well as the rules for transforming templates are driven by the YANG model to which the data is compliant. The limitations of the YANG modelling language also limit the TOC System's ability to transform the modelled data in certain, rather rare cases. For example, the data for a user-ordered list cannot be rendered from multiple source templates. The YANG models simply do not contain sufficient information to determine the order of data.

6. Discussion of Selected Use Cases

The following section provides examples of use cases to illustrate and highlight selected aspects of the proposed framework and benefits to Cable Operators.

6.1. Modular Data Design

One of the key advantages offered by the proposed templating methodology is the enablement of modular data design. The YANG-based data model of a MAC-NE can be divided into a set of small-scale segments, where each such segment corresponds to an integral functional area of the MAC-NE. The model break-up naturally follows the hierarchy of the MAC-NE YANG schema tree.

How might this work in practice?

The Cable Operator creates as many templates for each segment of the model as practically necessary to cover all variations of configurations needed for deployment and ongoing management. Several templates

providing configuration prototypes for defined deployment cases in a certain functional area can form a library of fundamental templates. An appropriately crafted set of fundamental template libraries can cover the entire data model of a device.

Figure 8 illustrates how such approach operates with an example based on three libraries of fundamental templates: the RF Subsystem, the Cable Bundle and the PTP Subsystem. Only three fundamental libraries are included in the example for brevity.

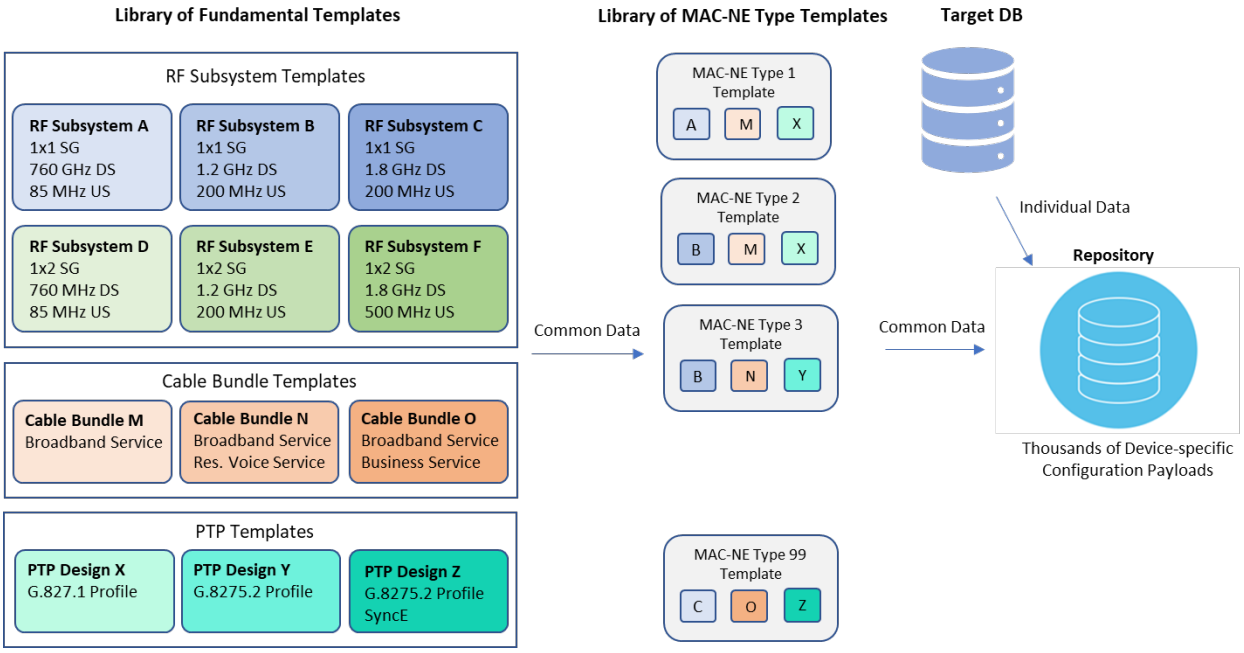

Figure 8 - Libraries of Fundamental Templates

The RF Subsystem template library includes templates for six designs of the HFC Plant. The PTP Subsystem template library includes three templates for commonly deployed PTP profiles. The Cable Bundle template library provides three templates for end-user IP service deployment options.

Next, from the library of fundamental templates, the TOC System renders a larger number of derived templates corresponding to as many variations of MAC-NEs as required. The intermediate library of MAC-NE Type templates is shown in the center of the diagram. Each MAC-NE Type template payload consists of selected payloads from the fundamental libraries.

In the subsequent step, the TOC System injects individual data into a selected MAC-NE Type template to create as many device-specific configuration payloads as needed. The device Configuration Data Payloads are made available to the MAC Manager to load onto the deployed RMDs.

6.2. Subdivision of Resposibilities within the Organization

Dividing work tasks between organizational teams is an essential part of running any business. This is particularly important for Cable Operators' organizations where separate operations sub-departments are often responsible for management of service and equipment lifecycle in functional areas under their jurisdiction. The examples of such functional areas include the DOCSIS® Subsystem, the MPEG Video Subsystem, Voice Services, Timing Operations or Subscriber Services.

The proposed framework, through the modularity in configuration data design, also enables the division of responsibilities between sub-departments within the Cable Operator's organization. The personnel of any department can maintain the sole ownership of the set of templates that correspond to the departmental responsibilities. These assignments can be enforced by effective access controls. For example, only the personnel of the department responsible for MPEG video service can be given the ability to modify (write-access to) the templates with configuration of MPEG video channels, DSG and out-of-band services.

6.3. Configuration Lifecycle Management

The flexibility and modularity offered by the Template-Oriented Configuration Management System simplifies many configuration lifecycle management tasks. One specific such task is the process of configuration modification within the template-oriented CMS.

Any changes to the configuration of a device due to network design changes, a software upgrade or service changes are typically contained within a small portion of the devices' configuration set. With proper template design, most configuration modifications typically affect only a single template, or a limited set of target variable values kept in the Target Database.

An administrator can modify a template by one of three methods.

1. The desired modifications are made directly to the payload of the affected template, resulting in a new revision of the template.
2. If the Repository already maintains a template with the desired configuration set, the name of the new template replaces the "old template" name in those Target Database records that need to be affected by the change.
3. The administrator creates the new template with a new name, often by cloning and modifying the previously used template. Then, the administrator modifies the Target Database as in Step 2 above.

After necessary modification to the templates and committing the changes to the Repository, the iterative rendering process predictably propagates the changes to the set of templates derived from the modified template and finally to the set of device Configuration Data Payloads.

7. Conclusion

In recent years the cable industry has embraced Distribute Access Architecture (DAA) including Flexible MAC Architecture (FMA) for innovation and significant investments. The paper augments these cable architecture transition efforts by examining the configuration aspects of DAA deployments and demonstrating how Cable Operators can effectively leverage YANG-based Configuration Template Methodology to scale, simplify and automate system configuration tasks within their back-office systems. The business outcomes are in the form of modular, automated, and streamlined provisioning systems and processes with increased agility and reduced OPEX.

Acknowledgements

We would like to sincerely thank our colleagues who made the writing of this paper possible, especially Brian Hedstrom of OAM Technology Consulting LLC for diligent review and mindful improvement suggestions.

Abbreviations

API	Application Programming Interface
BSS/OSS	Business Support System / Operations Support System
CCAP	Converged Cable Access Platform
CI/CD	Continuous Integration / Continuous Development
CLI	Command Line Interface
CMS	Configuration Management System
DSG	DOCSIS Set-top Gateway
FMA	Flexible MAC Architecture
IETF	Internet Engineering Task Force
JSON	Java Script Object Notation
MAC-NE	media access control network element
MMI	MAC Manager to MAC-NE Interface
MPEG	Motion Pictures Experts Group
OFDM	Orthogonal Frequency Division Multiplexing
OOB	out of band
PTP	Precision Time Protocol
RF	Radio Frequency
RFC	Request For Comments
RMD	Remote MAC Device
SC-QAM	Single Carrier Quadrature Amplitude Modulation
TMA	Template Management Application
TOC	Template-Oriented Configuration Management
VCS	Version Control System
XML	Extensible Markup Language
YAML	Originally: Yet Another Markup Language, recently: YAML Ain't Markup Language
YANG	Yet Another Next Generation
YCT	YANG-based Configuration Template

Bibliography & References

[FMA-MMI] CableLabs FMA MAC Manager Interface Specification, CM-SP-FMA-MMI-I03-220126.

[FMA-OSSI] CableLabs FMA OSS Interface Specification, CM-SP-FMA-OSSI-I02-220602.

[FMA-SYS] CableLabs Flexible MAC Architecture System Specification, CM-SP-FMA-SYS-I03-220126.

[RFC 6020] IETF RFC 6020, YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), October 2010.

[RFC 7950] IETF RFC 7950, The YANG 1.1 Data Modeling Language, August 2016.

[RFC 7951] IETF RFC 7951, JSON Encoding of Data Modeled with YANG, August 2016.

[RFC 7952] IETF RFC 7952, Defining and Using Metadata with YANG, August 2016.

[RFC 8040] IETF RFC 8040, RESTCONF Protocol, January 2017.

[RFC 9195] IETF RFC 9195, A File Format for YANG Instance Data, February 2022.

