Datacom Network Open Programmability V100R020C10

Product Overview

Issue 01

Date 2021-02-05

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

About This Document

Overview

This document describes the application scenarios and basic principles of the Open Programmability System (OPS), helping users quickly understand the product knowledge related to the OPS.

Intended Audience

This document is intended for:

- System administrators
- Maintenance engineers
- Technical support engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
DANGER	Indicates a hazard with a high level of risk which, if not avoided, will result in death or serious injury.
WARNING	Indicates a hazard with a medium level of risk which, if not avoided, could result in death or serious injury.
CAUTION	Indicates a hazard with a low level of risk which, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. **NOTICE** is used to address practices not related to personal injury.

Symbol	Description
NOTE	Supplements the important information in the main text. Note is used to address information not related to personal injury, equipment damage, or environment deterioration.

GUI Element Reference Conventions

The GUI element reference formatting that may be found in this document are defined as follows.

Format	Description
""	Buttons, menus, parameters, tabs, windows, and dialog titles are in **boldface**. For example, click **OK**.
>	Multi-level menus are in **boldface** and separated by the ">" signs. For example, choose **File** > **Create** > **Folder**.

Command Formatting Conventions

The command formatting that may be found in this document are defined as follows.

Format	Description
Boldface	Command keywords which must be reserved exactly are in **boldface**.
Italic	Command parameters which must be replaced by specific values in an actual command, are in *italics*.
[ ]	Items (keywords or arguments) in brackets [ ] are optional.
{x | y |...}	Indicates that one option is selected from two or more options.
[x | y | ...]	Indicates that one or no option is selected from two or more options.
{ x | y | ... } *	Indicates that multiple options are selected from two or more options. At least one option must be selected, and at most all options can be selected.
[x | y | ...] *	Indicates that multiple options are selected or none is selected from two or more options.

Change History

Issue	Date	Description
01	2020-08-30	This is the first official release.

Contents

1 Overview

1.1 Challenges in Network O&M

This section describes the challenges faced by carriers in network O&M.

Industry Trends

In the 5G era, network O&M is facing growing challenges. Network O&M is more like traffic operations as its value objectives are changing from lowering costs to generating benefits. Traditional network O&M uses a typical manual mode requiring both human efforts and process assistance. It is semi-manual and semi-automatic, and is evolving towards full automation. Network O&M evolution first requires a change in the traditional thinking pattern, that is, from separation of development and O&M to integration of them. To achieve this, the carriers' O&M department must have Development and Operations (DevOps) capabilities that can transform network O&M from traditional CT O&M into diversified ICT O&M.

In the era featuring rapid development of cloud computing, the design concept of using the cloud-native architecture to ensure cloud service experience as adopted for OTT services has a profound impact on the network construction mode of carriers. Regardless of whether to follow carriers' bottom-up practice of "network first, cloud later" or OTT service providers' top-down concept of "one network above cloud", the common factor driving the selection is that enterprises' digital transformation is accelerating the requirements for cloud-network synergy. Cloud-based network resource migration and cloud-network integration require both carriers and enterprises to offer on-demand customization, so as to support future service scenarios.

Customer Pain Points

In the traditional manual O&M model featuring "people + process", carriers are faced with the following pain points:

- Carrier networks are in the dilemma of multi-vendor device management. Using devices from multiple vendors is a long-term strategy for carriers and

enterprises to avoid being locked in. However, controllers of a single vendor can only manage its own network devices, and there is no unified interface standard for integration with the OSS system. The efficiency in adapting to a new device depends on the vendor's capability and response speed, which restricts the evolution in automatic provisioning of end-to-end network services. The adaptation efficiency remains slow for a long time and has become an industry-recognized bottleneck.

- It takes a long time — usually half a year, or even one to two years — to roll out a new service, which obviously cannot be able to meet the requirements of the new era. Among the numerous reasons for low rollout speeds, the typical one is that the development and O&M of new services are separated. That is, based on the new service requirements raised by carriers, device vendors develop and release versions, which will then be accepted and used by carriers. This means a long process.
- To complete tasks such as network device adaptation and network cutover, an engineer needs to manually execute a large number of command scripts, which is error-prone. As the scripts increase in scale, their maintainability keeps decreasing, making network O&M a high-risk task.

Obviously, this network management and control concept that centers on single-vendor configuration and baseline cannot meet carriers' increasingly flexible and agile O&M requirements. However, these requirements can be adequately addressed by multi-vendor-oriented open and programmable network management and control solutions.

1.2 Introduction to the OPS

To address the severe challenges in network O&M, the YANG-model-based OPS provides end-to-end open programming capabilities, including device driver programmability, network service programmability, open northbound interfaces (NBIs) for devices and services, as well as secure and reliable assurance mechanisms.

1.2.1 Capability

Device Capability Openness

With device capability openness, you can develop functions on the network element (NE) layer, manage devices, customize device functions, and enable device capability openness by customizing and loading device YANG models on NCE.

Device functions can be developed only by Huawei engineers currently; however, the functions will be released to third-party images for third-party development.

Service Capability Openness

Service capability openness allows you to customize and load service YANG models, service logic, and network service applications based on NCE, for example, L3VPN service model (IFTF-defined service northbound model). If network services are orchestrated across multiple devices, service configurations will be delivered to these devices concurrently.

Service applications can be developed by third parties or Huawei engineers.

Northbound Capability Openness

Northbound capability openness allows systems to automatically generate northbound RESTCONF, CLI, and web UI interfaces based on device or service YANG models to quickly interconnect with northbound man-machine and machine-machine interfaces.

1.2.2 Application Scenarios

This section describes the main application scenarios of the OPS.

Quick Adaptation to Multi-vendor Devices

Carrier and enterprise networks usually have devices from multiple vendors, typically in scenarios such as automatic 5G site densification and multi-vendor CPE configuration. However, these devices cannot be managed and controlled in a unified manner. In addition, slow integration of new devices, low automation, and long service provisioning periods have become bottlenecks in end-to-end service delivery. The OPS is a perfect choice for improving device adaptation efficiency. It can automatically identify and read YANG model files of devices through YANG interfaces, generate NE driver packages, and load the packages to the system. In this way, a new device can get managed within one day, improving the adaptation efficiency by 90%.

Rapid Service Rollout

New service rollout depends on the OSS system and controller version update. It takes a long time to roll out a new service due to problems such as insufficient API interface integration and high customization costs. Such a service rollout speed cannot support flexible service scenario changes. The OPS allows users to customize service YANG models and service logic, and can automatically generate northbound APIs to quickly integrate with the OSS system, so that users can add, delete, modify, and query devices and network services. The version development time is shortened from 6-9 months to only 1 month (agile and on-demand release), and the service rollout period is reduced by 80%.

Reliable Network Change Mechanisms

Legacy network O&M involves migration and modification of a large number of services. These changes are achieved by manual operations or command scripts, which are error-prone. It is difficult to maintain massive numbers of scripts, leading to high risks in network changes. To resolve these difficulties, the OPS provides secure and reliable mechanisms such as Dry-Run, rollback, transaction, and concurrency, improving the network change accuracy to 99.9%.

Full-Stack Programmability for Management, Control, and Analysis

In the 5G network slicing and intelligent O&M scenarios, the OPS not only implements programmable service provisioning, but also provides path computation and intelligent analysis programmability based on user-defined network service models, maximizing support for future network evolution of carrier services.

1.2.3 Feature Introduction

This section describes the main functions of the OPS.

Model Driven

- Service models and device models are developed based on YANG models.
- Service models and device models can automatically generate NBIs, including the CLI, RESTCONF, and web UI.
- Device models can automatically generate southbound protocol packets, including NETCONF protocol packets.
- Model-driven databases are supported. YANG models automatically generate database entries.

Dynamic Loading of Software Packages

By compiling and loading software packages, users can get new devices managed and new service provisioned quickly.

The system provides the following software packages:

- Specific NE Driver (SND) package: provides a data model for the interaction between the OPS and NEs. This data model usually contains a .py file and a YANG data model of several features. The .py file defines NE information, such as the device type, vendor, and connection information. The YANG data model describes the data structure of NE features. After an SND package is loaded to the system, the system can establish a connection with the device, query data, and deliver configurations to implement device management. The supported SND package types include NETCONF SND, CLI SND, NETCONF&CLI SND, RESTCONF SND, and Customized SND (provides the common SND package to enable YANG to support other protocols).
- Specific Service Plugin (SSP) package: defines a data model for completing a set of network-level service configurations. The data model usually contains a Jinja2 template file, a Python mapping script, and a service YANG model. Among these files:
  - The Jinja2 template describes the data structure of services and uses the Jinja2 syntax to perform operations such as interpolation, condition judgment, and recursion.
  - The Python mapping script describes how to fill in the template with the data submitted by users and map the data to the NE data structure.
  - The service YANG model describes service parameters and is constructed based on service input.

  After an SSP package is loaded to the system, the system can deliver service configurations to quickly provision new services.

Transaction Mechanism

The OPS provides a transaction mechanism, in which configuration changes can be committed in an atomic transaction, so that the data in the OPS is consistent with that in the forwarder. Data in a transaction is concurrently delivered to multiple devices. Either all the data is successfully delivered or all is rolled back, which means partial delivery of the data is not allowed.

Data Consistency

The OPS saves a copy of the data delivered to a device. The OPS can collect device data, discover differences between the device data and the data on itself, and display the differences on the GUI. If differences are detected, data synchronization can be performed based on the OPS data or the device data.

Open Programmability

- The OPS supports NBIs such as RESTCONF, and allows users to develop new capabilities based on Python scripts.
- The OPS supports model-driven programming interfaces. When compiling an SSP package, a user simply needs to write the creation process. The update and delete operations are automatically calculated by the EasyMap algorithm provided by the OPS. This simplifies programming and makes development easier.
- YANG models can be used to automatically generate southbound packets, improving driver development efficiency.

Configuration Preview

Before delivering the data configured on the OPS to devices, users can preview the configuration data, and view the packets to be delivered to the devices and the differences between the old and new configuration data.

Configuration Verification

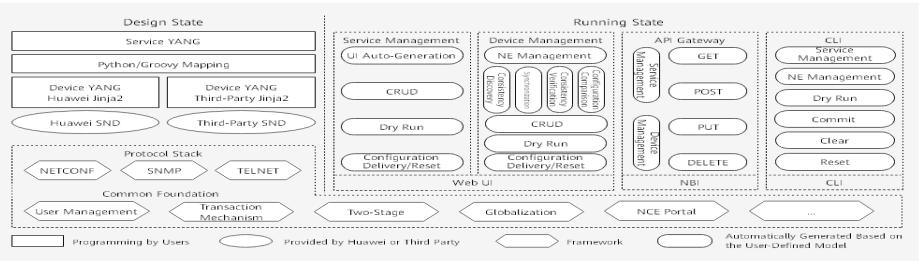
The data configured on the OPS can be verified based on the YANG model and the check syntax provided by the YANG model.

2 Fundamentals

2.1 OPS Architecture

The OPS includes the design state and running state. The design state is used to establish the mapping between service YANG models and device YANG models, whereas the running state is used to manage devices and provision services based on the mapping established in the design state.

Figure 2-1 OPS architecture



Contents in ▭ are provided by users, including service YANG models, device YANG models, and data mapping logic between the models.

Contents in ⬭ are SND packages generated based on the native YANG models of the devices. The SND packages can be provided by Huawei, users, or third parties.

Contents in ▢ are the CLI integrated on the web UI, as well as the service management UI, device management UI, and NBI automatically generated by the

OPS based on the user-defined service YANG models and device YANG models. Users can use these methods to manage devices and provision services.

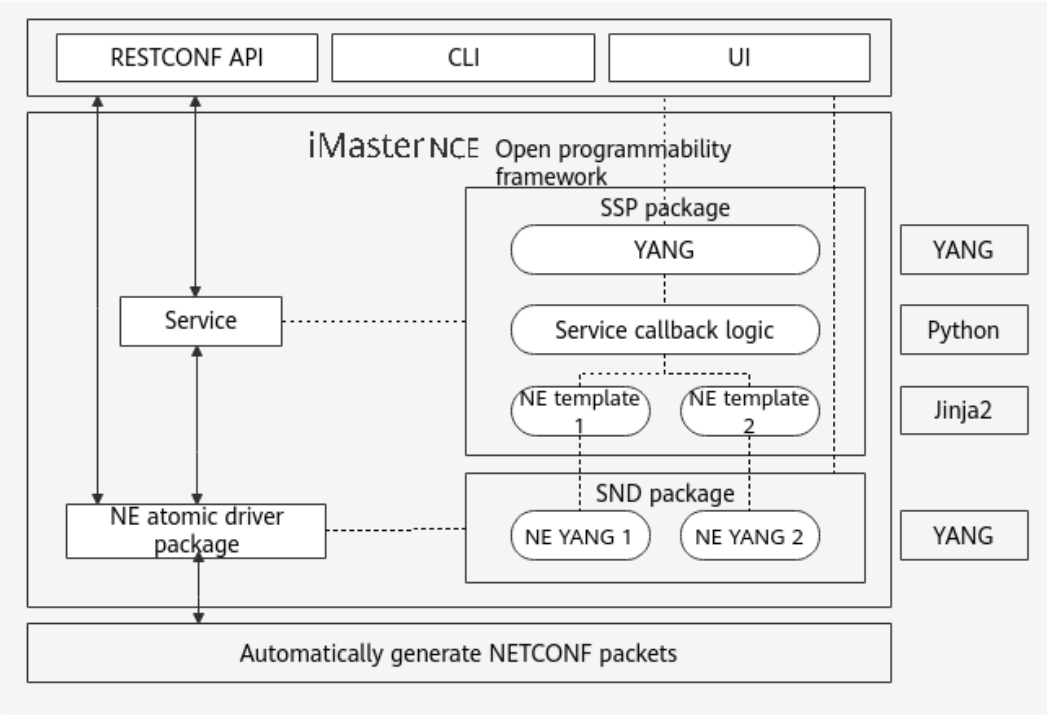
- The service management UI is generated based on service YANG models, and allows users to add, delete, modify, and query services based on the mapping between service and device YANG models.
- The device management UI is automatically generated based on device YANG models, and allows for data consistency verification, data synchronization, and configuration inconsistency elimination, as well as adding, deleting, modifying, and querying NE resources.
- RESTCONF interfaces are automatically generated based on service and device YANG models, and allow users to add, delete, modify, and query service and NE resources based on the mapping between service and device YANG models.
- The CLI is automatically generated based on service and device YANG models, and allows users to add, delete, modify, and query service and NE resources based on the mapping between service and device YANG models.
- The running state provides the Dry-Run function, which allows users to preview the results of the current operation and the modification of related device configurations.

2.2 Service Mapping Logic

Currently, the open programming framework supports two layers of mapping logic:

- Mapping from service models to device models using SSP packages
- Mapping from device models to protocol packages using SND packages

Figure 2-2 Service mapping logic



The detailed service mapping logic is shown in the right part of the preceding figure. The logic from top to bottom is as follows:

- Service YANG models automatically generate NBIs or configuration UIs.
- Users send configuration requests to their developed codes through the NBIs provided by service models.
- The service processing consists of two stages:
  - Python code processing: Vendor-independent service codes are processed at this stage, for example, a tunnel creation request. The generic service logic includes the tunnel path calculation logic.
  - Template code processing: Vendor-related codes are processed at this stage. A template is the data delivered to a device model, and varies according to vendors.
- SND packages are used to convert device models into protocol packets. For NETCONF-supporting devices, the OPS automatically converts device models into NETCONF protocol packets.

3 Specifications

Item		Specifications
Package repository management	Time required for dynamically loading a third-party Java package	≤ 120s
	Time required for uninstalling a third-party Java package	≤ 360s
	Time required for uploading a third-party package	≤ 30s
	Package repository capacity	• Maximum size of files extracted from a software package: 100 MB • SND package loading: A maximum of 200,000 leaf or leaflist nodes are supported. • SSP package loading: A maximum of 200,000 leaf or leaflist nodes are supported.
Template management	Time required for creating a template	≤ 2.5s **NOTE** In a template creation operation, the maximum size of template fields is 200,000 bytes.

Item		Specifications
NE management	Time required for establishing NE connections through NETCONF	60s (500 NEs)
	Device management capacity	• Number of devices managed through NETCONF: 30,000 devices by 3 nodes (One NETCONF connection is established for one device.) • Number of devices managed on the CLI: 5000 devices by 3 nodes (One connection is established for one device.) **NOTE** • A maximum of 500 devices can be imported at a time. • A maximum of 2000 devices can be added at a time.
Device configuration	Maximum number of concurrent configuration management transactions	200
	Number of concurrent batch configuration tasks	• Number of devices that can be configured at a time without delivering the configurations: 10 • Number of devices that can be configured at a time with the configurations delivered: 2000
	Maximum number of historical configuration records	100,000

Item		Specifications
Configuration consistency	Maximum number of devices for which concurrent configuration consistency operations can be performed (a single NCE node)	• Data synchronization: 20 southbound devices • Consistency verification: 20 southbound devices • Difference discovery: 20 southbound devices • Analysis: 3 service instances (including the number of service instances for which in-depth difference discovery is performed) • Difference check: 5 southbound devices

