FEP 2.0 Software Platform

White Paper

Copyright © 2026 TP-Link Systems Inc. All rights reserved.

No part of this document may be reproduced, copied, or transmitted in any form or by any means without the prior written permission of the company.

Contents

1 Overview 3
1.1 Introduction 3
1.2 Background 3
1.3 FEP 2.0: A Next-Generation Network Device Software Platform Solution 5
2 Key Technologies 7
2.1 Data-Driven Architecture 7
2.2 Linux Kernel and Hardware Isolation 8
2.3 Data Plane Rollback and Its Applications 8
2.4 Reliability Framework 9
2.5 Observability Framework 9
2.6 High-Performance Distributed Architecture 10
2.7 Full Omada Product-Line Compatibility 12
3 Typical Models 12
4 Appendix 12
4.1 Terminology 12

1 Overview

1.1 Introduction

In today's enterprise campus and data center networking solutions, reliability, scalability, high-performance services, and programmability have become fundamental requirements.

In these scenarios, end users expect seamless single-node service recovery, network-level self-healing capabilities, and high-performance high-availability features. Moreover, solutions are expected to support agile, efficient deployment and continuous, iterative service upgrades. Higher requirements are also placed on the solution's integrated controller in terms of information consolidation and automation.

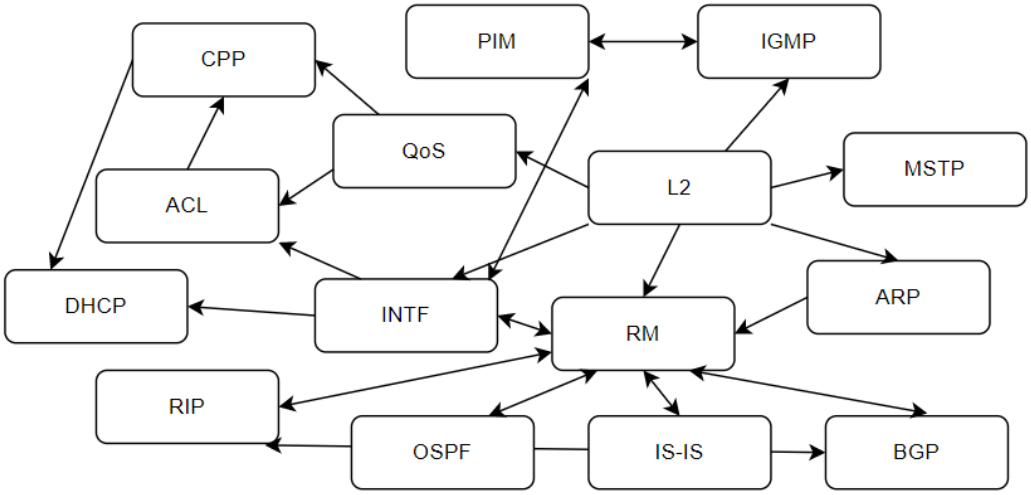
This document provides a detailed introduction to Flexible Enterprise Platform 2.0 (FEP 2.0), a unified software platform for modern networking, and explains how it supports campus-scale and larger networking requirements through multiple technical approaches.

1.2 Background

As industry demand for SDN networks and solutions continues to grow, device nodes in campus-scale and larger networking solutions have become increasingly feature-rich, and service interworking requirements have increased accordingly. From the perspective of software reliability, running services as independent processes is an inevitable choice, which introduces the need for Inter-Process Communication (IPC). In traditional network device software platform architectures, IPC is mainly implemented in two ways:

1. **RPC Procedure/function-call-driven software architecture: Remote Procedure Call (RPC)**

If we treat each service module process as a node and the procedure-call relationships between service modules as edges, we can draw the following RPC-based software architecture diagram:

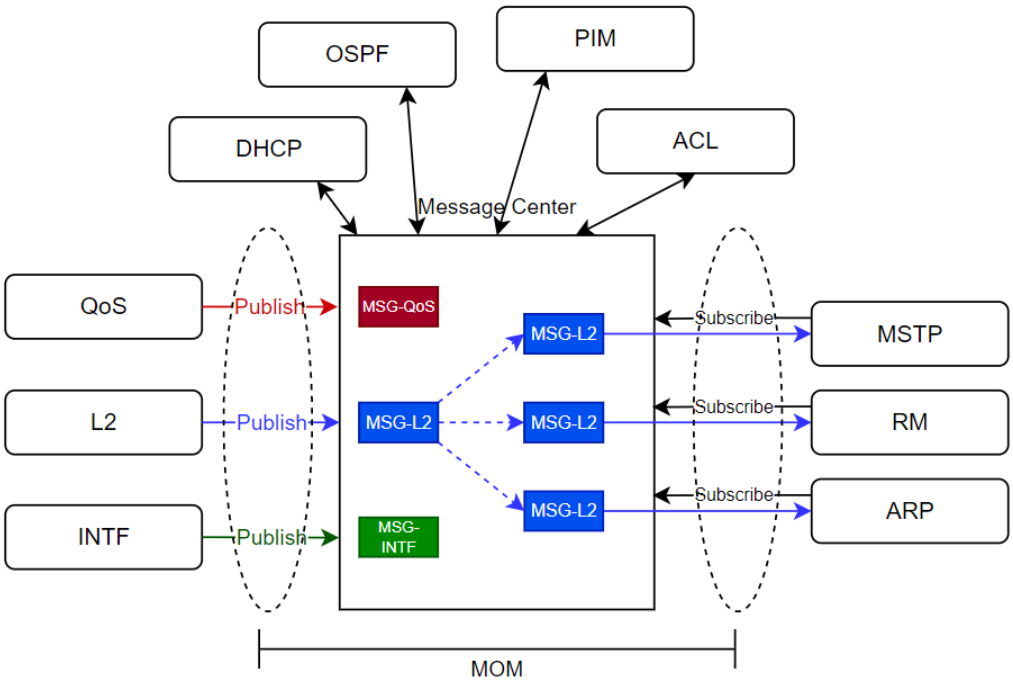


This architecture leads to a complex internal communication topology. Each user-facing service logic may form a long synchronous call chain involving multiple processes. Successful execution of the service logic depends on every process in the call chain waiting synchronously for the results, which reduces efficiency. A single-point service failure can propagate along the call chain. Long polling times or waiting leads to very low CPU utilization efficiency.

Additionally, as functional requirements increase, adding a new service process requires a considerable number of new communication channels. Existing channels must also be adjusted to ensure compatibility between the new service and existing services. In the RPC model, each communication channel typically corresponds to a unique interface, which introduces an unacceptable level of maintenance complexity and workload—disastrous for platform stability and scalability.

2. **Independent event-driven software architecture: Message-Oriented Middleware (MOM) and the Publish and Subscribe (PUB-SUB) model**

To avoid the systemic drawbacks of RPC, many network device software platforms adopt an independent event-driven architecture based on MOM and the PUB-SUB model. Such an architecture effectively addresses problems caused by tightly coupled service logic, including high maintenance burden, low reliability, and low performance:



As shown above, all services exchange data through MOM and the Message Center using independent events. Events carry necessary data and are routed within the Message Center, with event delivery implemented through the PUB-SUB mechanism. This approach comprehensively resolves the low performance, tight coupling, low scalability, and high maintenance complexity in the RPC model.

However, in large-scale mixed enterprise networks, service volume and data collection/analysis requirements continue to grow. As a result, this architecture must trade significant memory and CPU resources to handle resource shortages and performance issues caused by massive redundant messages accumulating under high load and frequent network disturbances.

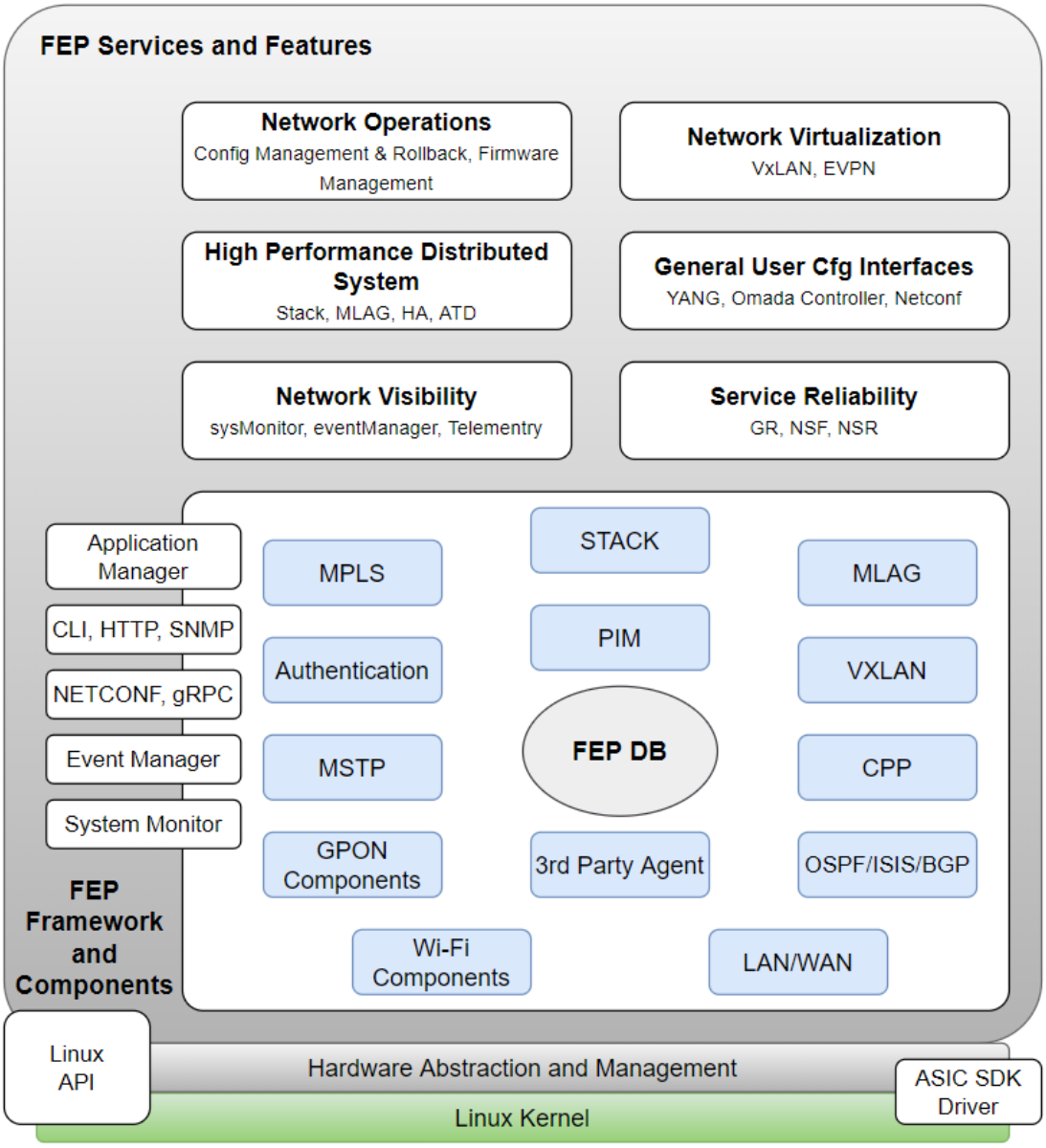
When consumers process messages more slowly than producers generate them, the Message Center queue can expand. Architects are forced into a dilemma: choose a limited-capacity queue and accept batch message loss or prolonged blocking; or choose a very large queue and increase memory and CPU costs to withstand high-load conditions.

1.3 FEP 2.0: A Next-Generation Network Device Software Platform Solution

FEP 2.0 addresses the pain points of traditional network systems. It aligns with the evolution of hybrid-cloud networking solutions and the rapid development of SDN networks, introducing multiple architectural designs and optimizations at the foundational layer.

This platform integrates modern application and operating system development concepts.

With support for multiple Linux kernel versions, it provides a fully decoupled three-layer IPC management framework, and offers the core **data-driven characteristic** using a high-performance standalone database. Building on this, the platform delivers transparent and elastic service restart capabilities, fully visible device network status, smooth service expansion and upgrades, and an open and unified user interface development model*. It achieves outstanding scalability in both service capabilities and product form factors.



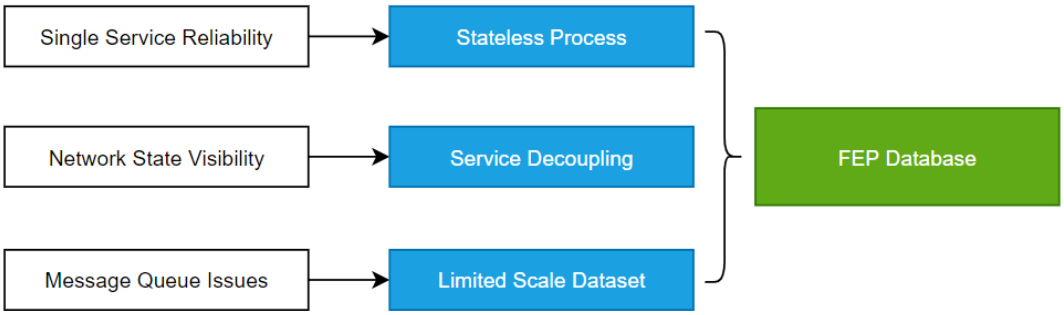
The next chapter introduces the core architecture components and key features of FEP 2.0 in detail.

Note: The open user interface development framework will be gradually enhanced as the FEP platform evolves.

2 Key Technologies

2.1 Data-Driven Architecture

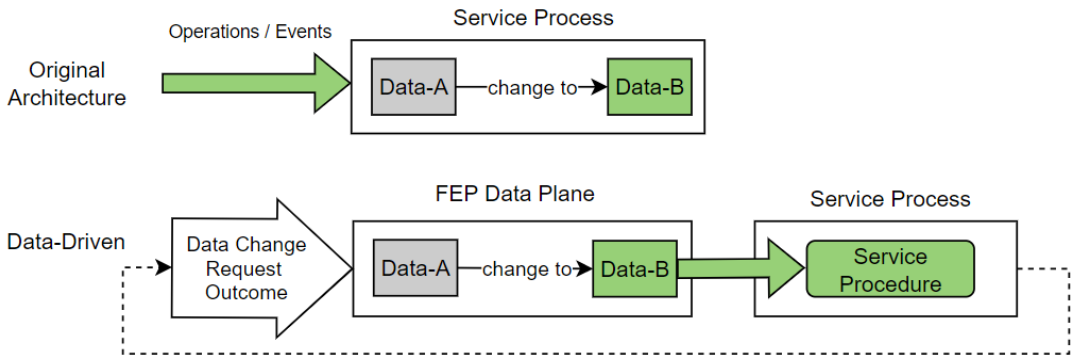
Data-driven architecture is a core feature of FEP 2.0. The requirements for it directly originate from the following platform-level needs:



FEP Database is an independent data plane. It is a bounded dataset that can proactively detect data changes and supports both the PUB-SUB model and a pull model. It can effectively perform change aggregation and change-event deduplication within the data plane based on consumer workload, thereby addressing the unbounded queue growth problem common in event-driven architectures.

An independent data plane means that device data publishing and presentation are fully decoupled from specific service processes. Failures in a single service are isolated within that process, and the device can quickly recover to its pre-failure state using historical data stored in the data plane.

A data-driven architecture provides network software platforms with exceptional system stability, comprehensive recovery capabilities, and extreme performance under high-load and complex network conditions. Under this architecture, the causal relationship of service logic also changes fundamentally:



- In traditional architectures, “operations” and “events” determine the service flow, generating new data.

- In data-driven architectures, "data changes" determine the service flow, generating new data.

2.2 Linux Kernel and Hardware Isolation

Omada series EAPs, gateways, and switches use different kernels and ASIC-SDK variants. FEP 2.0, through an independent Data Abstraction Layer (DAL) data plane and a virtual device management architecture, isolates Linux kernel functions (e.g., protocol stack), SDKs based on Linux kernel modules, and user-space SDKs from business processes. This enables better integration of the same software capabilities across different products or ASICs, and provides support for multiple reference and customized kernels.

2.3 Data Plane Rollback and Its Applications

Based on the data-driven architecture, FEP 2.0 provides a database that maintains the full device service state independently of service processes. Using two snapshots of **a specific subset of database data** at two points in time—DB(T1) and DB(T2)—the data transition from DB(T1) to DB(T2), along with the associated software and hardware state transition, is referred to as Data Rollback.

In traditional platform architectures, because DB(Ti) is distributed across multiple service processes, performing this type of full service-state transition typically relies on DB(T0), a fixed initial dataset, and requires a complex Reset & Load procedure involving nearly all services: DB(T1) → Reset → DB(T0) → Data-Loading → DB(T2). Regardless of how small the difference between DB(T1) and DB(T2) is—even if it is only a single CLI command—this process triggers Reset & Load across all service modules, greatly increasing the complexity of the rollback process and resulting in unacceptable performance overhead.

With the independent FEP database, an efficient direct diff mechanism between DB(T1) and DB(T2) is enabled: DIFF{ DB(T1), DB(T2) }.

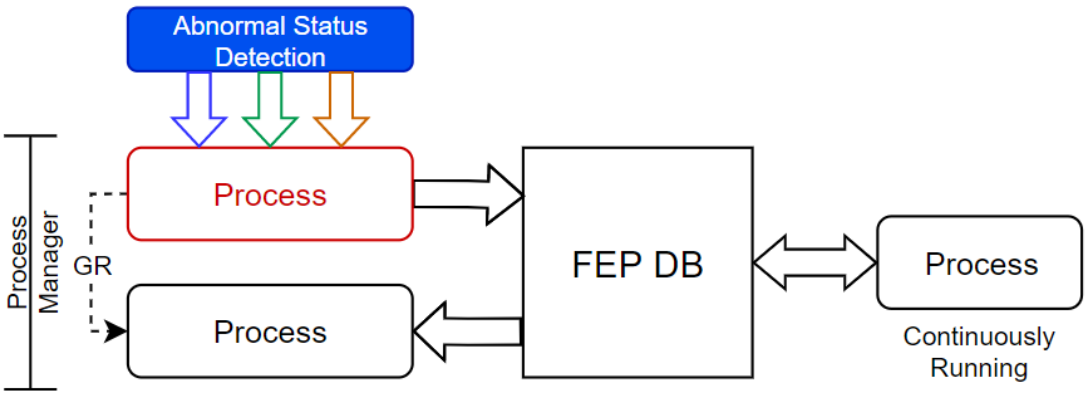
This mechanism directly identifies the state differences between two datasets and strictly confines the Data Rollback process to only the services where data changes actually occur, addressing the time and complexity issues of the Reset & Load.

Depending on how the subset DB(Ti) is selected, typical applications of Data Rollback in FEP 2.0 include:

- User Configuration Checkpoint Rollback
- Omada Controller Rapid Adoption
- Software/Hardware Consistency Check. Provides reconciliation and correction of software/hardware configuration and state

2.4 Reliability Framework

FEP 2.0 provides a process-based service reliability assurance framework:



Through a diverse and service-customizable set of ASDCs (Abnormal Status Detecting Components), the system detects abnormalities in service processes and triggers the Process GR (Graceful Restart) workflow when the abnormality level exceeds expectations.

With a fast-restart service process management framework and real-time independent backup of service state data, FEP 2.0 accurately detects exceptions and performs seamless graceful restarts without causing noticeable network disturbance.

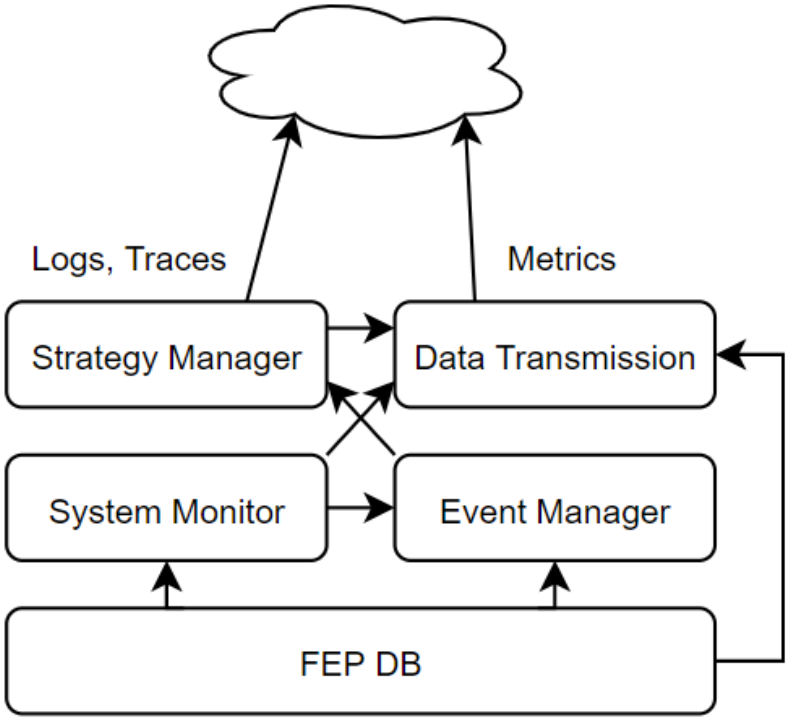
Faults in a single service are strictly contained within the corresponding service process and do not affect the FEP-DB data state, thereby improving overall system reliability.

2.5 Observability Framework

Observability of modern network devices is reflected in three types of information: Metrics, Logs, and Traces.

FEP 2.0 extracts this core data from within service processes through a unified observability framework and stores it in the FEP-DB. With high-performance data management middleware, system-wide data can be efficiently pushed, exported, and analyzed. With the Omada Controller, it enables comprehensive visualization of real-time status and health across hybrid networks, significantly improving solution-level deployment, monitoring, and troubleshooting capabilities.

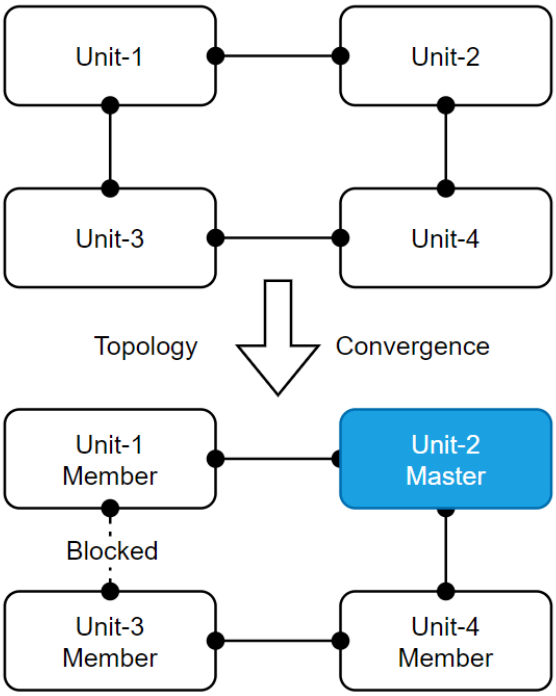
The observability framework includes four components: System Monitor, Event Manager, Strategy Manager, and Data Transmission.



2.6 High-Performance Distributed Architecture

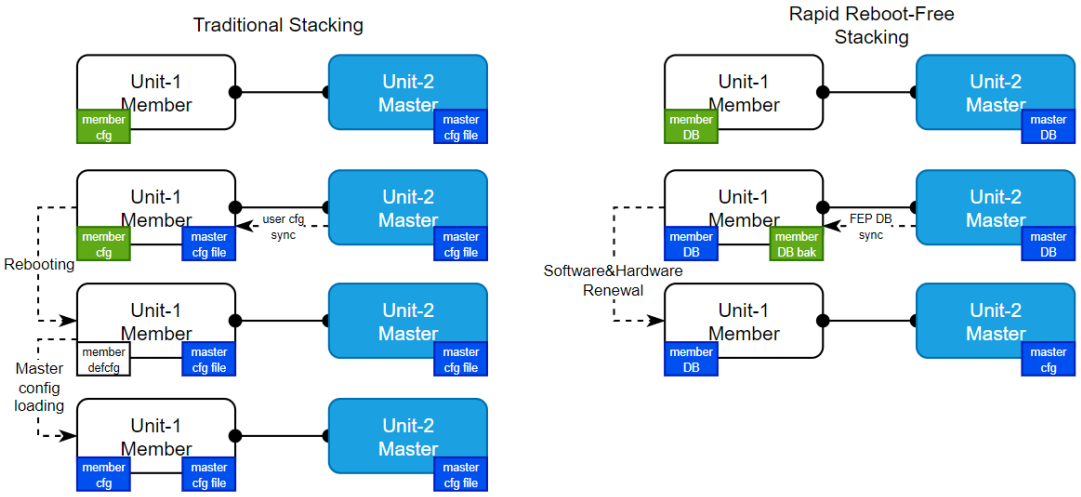
Distributed systems, featuring stacking and Multi-Chassis Link Aggregation Group (MLAG), play a critical role in smooth upgrades and scalability of networks built with multi-tier switches.

- Stacking virtualizes multiple physical devices into a single logical device with a unified control plane.
- MLAG is a lightweight virtualization of the forwarding plane. It supports only two devices and maintains independent control planes.



When building a stack topology, configuration data must be synchronized among devices, and the hardware state of the receiving device must be consistent with the data provider partially or fully. In traditional architectures, this typically requires rebooting the receiving device so it can transition from the factory state to the target state consisting of the primary device via configuration loading. On campus network devices, initialization during reboots may take several minutes, which is a key bottleneck affecting the stacking efficiency and initial deployment experience.

In FEP 2.0, stacking also leverages data-driven architecture. Through the FEP-DB's cross-device data synchronization mechanism, it performs a unified switch of the distributed system data plane and enables real-time software and hardware updates, achieving fast stack convergence.



2.7 Full Omada Product-Line Compatibility

With a highly flexible, customizable IPC architecture, FEP 2.0 extends the "shared platform" concept from switches and gateways to EAPs and OLTs while flexibly balancing single-node reliability and observability, service scale and performance, and hardware resource cost to meet product requirements. It truly enables a shared platform across campus-scale and larger devices and SMB access-layer devices, achieving full Omada product-line compatibility. This fully leverages the shared-platform advantages and delivers consistent, highly reliable, high-performance, and uniform solution-level user experience.

3 Typical Models

All Omada product lines will migrate to FEP 2.0 according to the following schedule:

- Omada Campus/Aggregation/Access Switches are expected to complete the migration by H1 2027.
- Omada Wi-Fi 7 APs are expected to complete the migration by H2 2027.
- For OLT products, DS-P8000-X7 already supports FEP 2.0, and DS-P8000-X2 is expected to migrate to FEP 2.0 in December 2026.

4 Appendix

4.1 Terminology

Term	Explanation
ASIC	Application Specific Integrated Circuit
ASDC	Abnormal Status Detecting Component
Data-Driven	Data-Driven (service processes are driven by data changes)
FEP	Flexible Enterprise Platform
IPC	Inter-Process Communication
MLAG	Multi-Chassis Link Aggregation Group
MOM	Message-Oriented Middleware
Process GR	Process Graceful Restart (fast restart with no service impact)
PUB-SUB	Publish and Subscribe
RPC	Remote Procedure Call
SDK	Software Development Kit
SDN	Software-Defined Networking
SMB	Small/Medium Business