

Architecture Foundations and Deployment Scalability for CWMP: A Technical Review

Marco Marinello(✉)

Qnets, "Internet that Just Works", 20143 Milano, MI, Italy
marco@marinello.bz.it

Abstract. The complexity of the Internet requires the use of sophisticated management protocols capable of handling heterogeneous hardware at different functional levels. The CPE WAN Management Protocol (CWMP) serves as the basic framework for the management of subscriber devices from afar, emerging from the initial TR-069 specification to the more complex data model in TR-181. This paper provides an overview of the architecture behind CWMP, specifically focusing on the importance of separating the SOAP-based communication infrastructure from the data models themselves, enabling the management of various hardware devices, such as home gateways, Access Points, and VoIP phones. Unlike existing literature that focuses primarily on theoretical scalability or high-level autonomic loops, this paper contributes a practical analysis of "zero-touch" lifecycle management within high-concurrence ISP environments. We examine the deployment of these protocols through the lens of GenieACS, an open-source implementation that leverages TypeScript and MongoDB to handle high-concurrency ISP workloads. The originality of this work lies in its systematic review of modular ACS design and its identification of the critical documentation and implementation gaps that hinder the adoption of Free and Open Source Software (FOSS) solutions in production. Finally, we discuss the transition toward emerging methodologies, such as the User Services Platform (USP/TR-369), in the context of the lessons learned from two decades of TR-069 lifecycle management.

Keywords: TR-069 · CWMP · GenieACS · Device Management · Zero-Touch Provisioning · Internet · ISP Operations · FOSS

1 Introduction

The evolution of the Internet from a collection of static workstations to a massive ecosystem of billions of heterogeneous endpoints has fundamentally reshaped network management. In the contemporary Internet of Things (IoT) and broadband landscape, Customer Premises Equipment (CPE) – ranging from high-speed home gateways to smart utility meters – is increasingly managed externally by service providers. This "zero-touch" paradigm requires seamless remote provisioning, configuration, and maintenance without end-user intervention.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16767, pp. 192–203, 2027.
https://doi.org/10.1007/978-3-032-30491-9_13

To address the operational impossibility of manual configuration at scale, the Broadband Forum[1] introduced the CPE WAN Management Protocol (CWMP) [1], designated as TR-069[2]. By defining a standardized communication layer over HTTP/SOAP between the CPE and an Auto Configuration Server (ACS), TR-069 provides a robust framework for firmware management, status monitoring, and diagnostic execution.

While being one of the most widely used solutions in global telecommunications infrastructure, implementation techniques, as well as operational peculiarities of TR-069-based ACS are not widely discussed in the literature, where more modern approaches like MQTT and CoAP take center stage. Moreover, many previous works often neglect the challenges associated with implementation of ACS in a high concurrence environment. This paper aims to fill that void through a systematic review of the structure of TR-069-based ACS deployments. In particular, we analyze the process of CWMP session initialization, explore key features of the RPC mechanism used for managing the devices, and present a modular design of ACS implementation architecture.

2 Related Work

Research into remote device management has grown since TR-069 appeared. However, most papers focus on high-level architecture or network theory, often skipping the messy details of protocol implementation. Many studies prioritize mathematical scalability models for large carrier networks [2]. These help define what autoconfiguration servers can theoretically do, but they usually ignore the technical friction found in real-world operations.

Rachidi and Karmouch [3] offer one example of this theoretical focus. They designed autonomic devices using IBM's MAPE-K loop, showing how CWMP can reconfigure Set-Top Boxes when quality drops. Their work clarifies autonomic logic but largely overlooks the protocol's actual constraints.

Stusek et al. [4] broadened the scope by applying TR-069 to smart meters and health monitors. While their early work stayed abstract, their extended study [5] introduced a modular implementation using OSGi bundles and GenieACS. This shift from theory to a functional smart home gateway architecture highlights the protocol's practical utility in heterogeneous IoT environments.

Our study bridges theoretical design and operational practice. We analyze the ACS architecture through a strictly implementation-oriented lens. Instead of abstract models, we focus on session initiation and RPC management within GenieACS. This technical decomposition draws directly from experience managing large-scale production fleets. It addresses the practical friction points that theoretical papers often overlook.

[1] https://www.broadband-forum.org, accessed 8 May 2026.
[2] https://www.broadband-forum.org/pdfs/tr-069-1-6-1.pdf, accessed 8 May 2026.

3 Technical Architecture and Protocol Stack

CWMP is essentially a two-way protocol based on SOAP/HTTP that aims at facilitating communication between the Customer Premises Equipment and Auto Configuration Server. The following subsections discuss the architecture and protocol stack of the management framework.

3.1 Communication Stack and SOAP Encapsulation

The CWMP protocol stack relies on standard Internet protocols that ensure interoperability across diverse network hardware. At the transport layer, TCP/IP provides reliable delivery, while HTTP/S acts as the application carrier. TR-069 utilizes SOAP[3] to encapsulate management commands and responses in XML format.

A typical session consists of an HTTP Post request containing a SOAP envelope. This overhead, while significant for low-power IoT devices, allows for a highly structured and extensible command set, enabling the ACS to manipulate complex data models like TR-181.

3.2 Decoupling Protocol Logic from the Data Model

The primary strength of the TR-069 framework lies in the strict separation between the transport mechanism (CWMP) and the information payload. By keeping the SOAP/HTTP envelope agnostic to the XML content it carries, the protocol supports a data model that is both fluid and highly extensible. This abstraction ensures the management interface remains remarkably versatile, adapting not only to diverse device categories – such as VoIP gateways and DSL modems – but also to the incremental feature sets introduced across different firmware versions of the same hardware.

This flexibility is operationalised through standardised root objects, most notably `Device.` and the legacy `InternetGatewayDevice.` framework. When a manufacturer introduces new functionality via a firmware update, they simply extend the existing data model tree with new parameter paths. Crucially, the ACS does not require a protocol-level upgrade to support these additions; it merely needs to be aware of the new nodes within the hierarchy. Such inherent forward compatibility allows the ACS to function as a universal controller, managing a heterogeneous fleet of legacy and next-generation hardware through a single, consistent interface.

However, this extensibility often leads to a proliferation of vendor-specific parameters that diverge from the standard TR-181 definitions. Manufacturers frequently use custom paths to expose proprietary features or hardware-specific configurations or parameters. A query conducted on a production GenieACS

[3] Simple Object Access Protocol https://www.w3.org/TR/2000/NOTE-SOAP-20000508/, accessed 8 May 2026.

instance reveals the scale of this fragmentation. For instance, a TP-Link appliance contains 259 custom parameters, representing 18% of its total data model. In contrast, a ZTE device holds 491 (25%), while Huawei hardware reaches a significant 1127 (56%). Other vendors like AVM and Genexis show 767 (39%) and 61 (4%) custom parameters respectively. These variations, illustrated in Figs. 1, 2, 3, 4, and 5, force the ACS to handle thousands of unique paths that are not documented in the official TR-069 specifications.

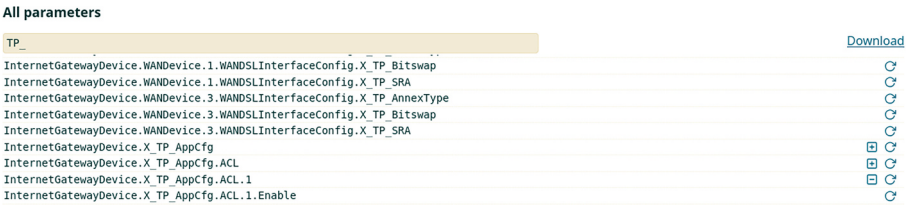

Fig. 1. Custom parameters: TP-Link

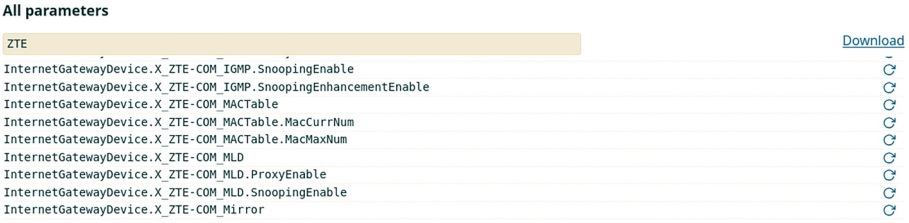

Fig. 2. Custom parameters: ZTE

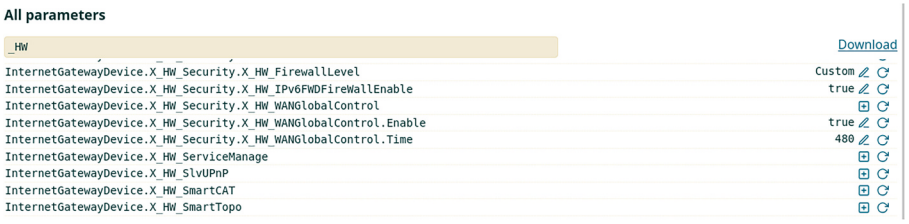

Fig. 3. Custom parameters: Huawei

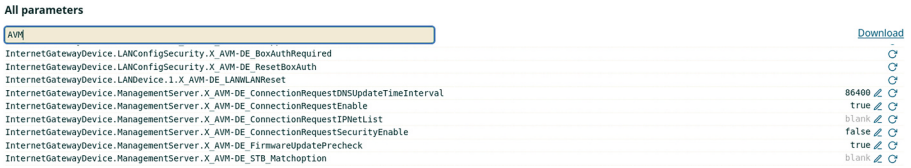

Fig. 4. Custom parameters: AVM

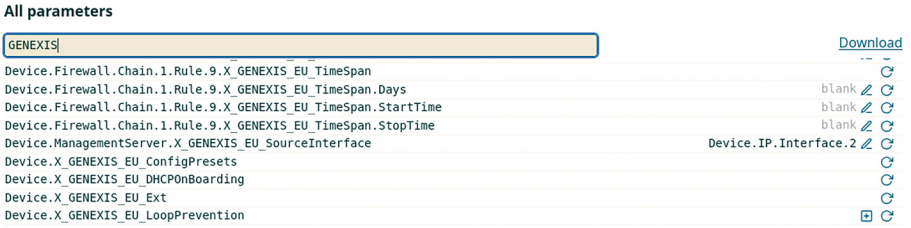

Fig. 5. Custom parameters: Genexis

3.3 Security and CPE Authentication

The security architecture of TR-069 is designed to ensure both data confidentiality and the authenticity of the managing entities, particularly when operating over public wide-area networks. At the transport layer, the protocol relies on TLS[4] to establish an encrypted tunnel. This prevents the interception of sensitive configuration parameters – such as Wi-Fi, PPPoE or SIP credentials – which would otherwise be transmitted in plain-text XML within the SOAP envelope.

Beyond transport encryption, the protocol implements an identity verification process via HTTP Digest Authentication[5]. A critical aspect of the CWMP lifecycle is the bidirectional nature of this trust: the ACS must authenticate the CPE to ensure it is managing, and sending sensitive data to, a legitimate device, while the CPE must authenticate the ACS to prevent "rogue server" attacks.

3.4 Session Lifecycle and Inform Mechanism

The operational lifecycle of a CWMP session is fundamentally "CPE-initiated", a design choice intended to navigate the complexities of NAT[6] both at a tenant-level, to manage devices behind the main router and, at a carrier level, for ISPs that implement CGNAT[7].

[4] Transport Layer Security – RFC 8446 https://datatracker.ietf.org/doc/html/rfc8446, accessed 8 May 2026.
[5] RFC 7617 (obsoletes 2617) https://datatracker.ietf.org/doc/html/rfc7617, accessed 8 May 2026.
[6] Network Address Translation.
[7] Carrier Grade NAT – RFC 6888 https://datatracker.ietf.org/doc/html/rfc6888, accessed 8 May 2026.

Every interaction between the management system and the device begins when the device sends an HTTP POST request to the ACS. The message serves as a state report that includes the device's identity[8] along with a set of "Event codes" indicating why the connection was established.

3.5 Remote Procedure Calls and Managed Download Process

On the functional layer, TR-069 leverages a well-defined collection of Remote Procedure Calls (RPCs), which should be used in order to change the parameters' tree of the device. The usual status polling and dynamic configuration are performed via `GetParameterValues` and `SetParameterValues` calls. However, Download is used as the primary way of performing more significant lifecycle changes. Unlike the simple change of a single parameter, a download operation is a stateful process that involves several stages and is tuned to transfer large volumes of information, such as firmware images or complex configuration blocks.

The effectiveness of this process is guaranteed by its asynchronous nature: in contrast with the conventional transfer method, where ACS would need to embed the file itself into the SOAP message, it uses another technique: it supplies the target CPE with the URI and authentication details. This allows CPE to accept the instruction and close the management session before initiating the file retrieval process on its own. In turn, the endpoint of the whole process occurs when the CPE finishes validation of the firmware and enters its new state; this fact is reported to the ACS with the means of a special `TransferComplete` RPC call. Offloading the actual content transfer part onto some external servers or even CDN[9] helps keep the ACS lean as the mere controller of this procedure.

4 Deployment Architecture and Implementation

While the CWMP protocol provides the guidelines on how communication should be conducted, the actual implementation of an ACS requires a powerful software architecture that is able to process highly concurrent requests. In this case, this paper focuses on GenieACS[10], as it represents the only existing FOSS solution that is currently being developed and used by the community.

As an open-source project, GenieACS provides a powerful engine but suffers from a fragmented documentation landscape. The official documentation[11] is sparse; it totals roughly 51 pages. Critical sections are incomplete. For instance, the "Roles and permissions" page contains only a "TODO" placeholder. This forces engineers to rely on the community forum[12] or tribal knowledge found

[8] defined by the Manufacturer, typically the device model, serial number, firmware version, etc.
[9] Content Delivery Network.
[10] https://genieacs.com/, accessed 8 May 2026.
[11] https://docs.genieacs.com/en/latest/, accessed 10 May 2026.
[12] https://forum.genieacs.com/, accessed 8 May 2026.

in the project's GitHub Wiki[13]. While the code is highly scalable, this lack of formal guidance remains the primary hurdle for production-grade adoption.

4.1 Architecture for Implementation and Deployment

GenieACS architecture represents the latest designs for ACSs, moving away from the monolithic systems to become a modular system based on services. The most critical component of the system is its being written in TypeScript, which provides the opportunity to apply strong type checking necessary for dealing with the complicated and often uncertain XML messages used by CWMP. Using the asynchronous nature of Node.js, GenieACS manages to process many management sessions at once without making closing SOAP connections a bottleneck.

Operational intelligence on the platform involves several specialized operations where the CWMP engine becomes the most significant one. CWMP engine acts as the protocol interface controlling the lifecycle of the session by monitoring and facilitating state transitions. The protocol interface receives `Inform` messages that need interpretation, and CWMP acknowledges them. Unlike many other software implementations, GenieACS decouples its protocol logic from the underlying storage. In this regard, GenieACS uses MongoDB as the main database for storing data. Such an option allows for accommodating different Data Models within one server. Since the parameters of devices represent trees (e.g., TR-181), using a document-oriented database like MongoDB allows storing such hierarchical data as JSON objects. This approach eliminates the "impedance mismatch", which occurs when using SQL databases. This way, ACS can handle and store different parameters from other manufacturers without a fixed structure.

Interactions with the system can further be categorized into two separate interfaces, which are referred to as the User Interface (UI) and the Northbound Interface (NBI). The User Interface is meant to provide a visual dashboard that is supposed to be used by administrators in diagnosing problems and taking manual actions. On the other hand, the NBI represents the functional part of ISP connectivity. As a RESTful interface, the NBI allows external logic such as billing services to control the ACS.

Finally, the FS module functions as a special storage place for the "Data Plane" of management, containing all of the firmware binaries and configuration blobs which are served through the Download RPC API. This particular module, as stated in 3.5, may be swapped out for any other FTP/HTTP server or even for some custom services. These modules provide a loosely coupled architecture in which all layers may scale independently, thus allowing the support of large-scale production network deployments.

[13] https://github.com/genieacs/genieacs/wiki/Example-of-a-Provisioning-Flow, accessed 8 May 2026.

4.2 Programmable Logic via Provisioning Scripts

GenieACS handles the complexity of heterogeneous device fleets through its scripting engine. This system lets engineers write TypeScript code, called "Provisions", that the ACS executes during each management session. Instead of relying on static configuration templates, the engine evaluates logic in real-time. Engineers define specific parameters and conditions to trigger these Provisions. For example, a Provision might only activate if the device belongs to a specific manufacturer or if a particular data model value is missing. This conditional execution ensures the ACS doesn't waste cycles on irrelevant tasks: a script for VoIP settings is triggered only if the device actually reports VoIP-capable hardware. By using TypeScript, the platform gives developers a familiar environment with strong typing. This helps prevent runtime errors that could brick a hardware unit. Such programmability enables "zero-touch" automation at scale. The ACS adapts its behavior based on what the CPE reports, handling vendor-specific quirks without needing a hardcoded core for every new hardware model.

4.3 Network Topology and Security Considerations

The location of the ACS in relation to the provider's network infrastructure can affect the effectiveness of the protocol and its vulnerability to attacks. In contemporary telecommunication networks, two main strategies are widely adopted. One of the most popular deployment schemes involves locating the ACS in a private management network owned by the carrier, which is located beyond CGNAT. Such a strategy shields the ACS from volumetric attacks but presents difficulties in terms of communicating commands to the Devices, since the ACS must have the proper routes and firewall permissions to communicate with them, without the need to wait for an `Inform`.

The ACS can be put online to enable device management within various network boundaries as well. Although it allows for greater flexibility, it necessitates the application of a zero-trust model for the protocol's security.

No matter what the topology of the network is, the integrity of the CWMP communication process will always be vital. Since the ACS acts as a central point of control, it becomes a highly valuable target for identity spoofing. One weakness that exists in the system is when the ACS only uses the serial number or OUI of the device to validate its identity. Without mutual authentication using TLS with certificate-based authentication or, at least, HTTP Digest Authentication, the attacker can easily spoof the identity of a legitimate device.

These implications can be quite severe. For one thing, a successful attack can give the intruder access to the full configuration payload that is stored on the targeted device. In many cases of residential deployment, the configuration payload contains confidential SIP trunking data used for VoIP communication or the passwords to access the gateway device. Apart from breaching the confidentiality of the individual, it is possible that a successful attack of this sort will also enable the attacker to gain access to the network infrastructure of the service provider itself. Consequently, the deployment of an ACS in a production

environment necessitates a rigorous security posture where the authenticity of both the CPE and the ACS are validated before any sensitive parameters are exchanged.

5 Lifecycle Management: from Zero-Touch Provisioning to Operational Maintenance

5.1 ACS Discovery and Bootstrapping Mechanisms

For a "zero-touch" deployment to succeed, the CPE must be able to resolve the ACS URI without manual configuration. In production environments, three primary mechanisms are utilized to "inform" the CPE of its management destination:

1. Factory-preset configurations: in many vertically integrated ISP models, the CPE is shipped with a hardcoded ACS URL embedded within the default firmware. While this is the most seamless method, it lacks flexibility, as migrating the device fleet to a new ACS infrastructure would require a coordinated firmware update or complex DNS redirection at the network level.
2. DHCP option 43[14] and PPPoE extensions: a more dynamic approach involves utilizing the network's access infrastructure to provide management metadata. During the initial IP address acquisition, the DHCP server can include option 43, which carries the ACS URL. Similarly, in DSL environments, vendor-specific tags within PPPoE discovery packets can serve the same purpose. This method is highly effective for "open-market" devices, as the management authority is determined by the network to which the device connects rather than the hardware's origin.
3. Manual intervention: while counter-intuitive to the goal of automation, manual entry remains a necessary fallback. This typically occurs via the built-in web management interface after logging in with elevated credentials. This method is often the only choice for small-scale businesses who cannot afford either the firmware personalization by the vendor nor direct control over the network.

5.2 ACS Admission Control and Device Adoption

The transition from a device "calling home" to being actively managed is governed by the ACS's admission control policy. In a production environment, the server must decide whether to treat an incoming `Inform` request as a legitimate registration or a potential intrusion. This is often managed through a choice between automated discovery and pre-authorized orchestration.

In the most flexible deployment models, the ACS acts as a dynamic listener, adopting any device that successfully negotiates the HTTP Basic or Digest

[14] DHCP Options and BOOTP Vendor Extensions – RFC 2312 https://datatracker.ietf.org/doc/html/rfc2132, accessed 8 May 2026.

Authentication handshake. While this "open enrollment" simplifies logistics – especially in high-volume consumer rollouts – it relies heavily on the integrity of shared secrets. Should these credentials be compromised or reverse-engineered from the firmware, the ACS becomes susceptible to "shadow" devices that could drain system resources or probe for management vulnerabilities.

To mitigate these risks, many operators moved toward a more rigorous pre-provisioning model, where the ACS is instructed to reject any CPE not explicitly registered in its database. This creates a "closed-loop" ecosystem where the device's hardware identity[15] must be white-listed before the first contact occurs. In this architecture, the GenieACS NBI becomes the primary facilitator of trust. Rather than the device initiating the relationship, the operator's CRM/ERP or billing system "introduces" the device to the ACS via a RESTful API call at the moment of subscriber activation. If an unlisted device attempts to initiate a session, the ACS rejects the request, refusing to proceed with the CWMP handshake. This integration ensures that the management plane remains strictly coupled to the business logic, guaranteeing that sensitive configuration data are reserved exclusively for verified assets.

5.3 First Provisioning

Following a successful adoption, the ACS must transform the CPE from a generic hardware state into a service-ready device. This initial provisioning phase is rarely a single-parameter push; rather, it is a nuanced synthesis of two distinct data sets. The server relies on the scripting engine (Sect. 4.2) to execute the logic required for this transition. First, the ACS applies ISP-defined global configurations, such as standardized firewall rules, DNS server assignments, and firmware baseline requirements. Second, it injects user-specific parameters retrieved from the CRM, which can include the subscriber's specific WAN configuration, Wi-Fi localization and personalization, and SIP trunking details for voice services.

In many high-security or complex carrier environments, this provisioning is handled through a dual-infrastructure handover. In this model, the device first contacts a "Provisioning ACS" located within a restricted, local provisioning network. This initial server's sole purpose is to perform the base-lining of the device, ensuring it has the minimum required firmware, security certificates and configuration. Once this internal stage is validated, the Provisioning ACS pushes a final command to update the `ManagementServer.URL` parameter, effectively handing over the CPE to the Production ACS. Considerations about the security of the Production ACS made in 4.3 remain valid.

5.4 Remote Diagnostics and Orchestration

The longevity value of a TR-069 implementation lies in the fact that after a one-time setup, it moves into a continuous period of maintenance and reactiveness. This interaction is maintained via periodic `Inform`, which is essentially the

[15] specifically its serial number and OUI.

heartbeat of the whole process that guarantees the ACS stays aware of what is happening on the device side. In this regard, forcing the CPE to check-in periodically gives MSP a longitudinal view of network status, thus providing the ACS the capability to act as an oracle and orchestrator of all activities on the device.

As for problem resolution, such check-ins provide the necessary information for resolving customer complaints without having to make an actual physical trip to their location. If a customer complains about slow connection, the ACS can refer back to previous telemetry collected during the check-in, including the noise margin of DSL lines, wireless interference levels, and high memory usage. This allow technicians to differentiate between a localized hardware failure and a broader environmental issue.

However, the effectiveness of these diagnostics is often hampered by vendor-specific implementation quirks. For example, while standardized RPCs for diagnostics exist, some manufacturers only expose a subset of the results, or worse, report a successful test even when hardware faults are present. Others may use proprietary parameter paths for standard metrics like optical signal strength (RSSI), forcing the ACS to maintain complex mapping tables for each vendor. Furthermore, the ACS can trigger real-time diagnostic RPCs, such as SelfTest-Diagnostics, effectively "virtually presence" the technician inside the customer's LAN to pinpoint bottlenecks in the service delivery chain.

Beyond individual support, the ACS manages the orchestrated evolution of the entire fleet through firmware and configuration roll-outs. Managing a mass upgrade is a delicate balancing act; pushing a new firmware to a hundred thousand devices simultaneously would create a thundering herd effect, potentially crippling the carrier's bandwidth and the ACS's own processing capacity. Another benefit of a slow roll-out is the increased chance to find peculiar configuration that causes the upgrade to fail and/or cause a service disruption. Deploying to uncorrelated setups helps testing the reliability of the new configuration and fix – if any – related issues. GenieACS addresses this by allowing operators to schedule these transitions across a distributed timeline. The ACS monitors the success of these campaigns through `TransferComplete` events, ensuring that any device failing the upgrade is flagged for manual review rather than being left in an insecure or non-functional state. This ensures that the network maintains its flexibility to evolve past initial factory settings.

6 Conclusions

From the TR-069 standards up to the current state of affairs, there have been significant changes in scope, as reflected in the increase in data models from TR-098 to TR-181, among other developments. Whereas this development allows for the increasing complexity of hardware being controlled, it brings about a considerable level of interoperability issues. CWMP may be considered a standardized technology, but its implementation in real-world scenarios depends on a lot of lab testing. There are specific vendor parameters and behaviours that do not conform to standards, which require thorough preproduction testing for every piece

of hardware or firmware update since there could be variations under the hood despite the existence of standards.

Nonetheless, it is through the adaptability of CWMP that it has found itself rising as the default engine behind the wide array of business management systems. The application has uses that extend beyond the traditional residential gateway to include managed Wi-Fi hotspots, Voice over IP phones, and many other kinds of devices. In this regard, GenieACS offers a modular, open-source alternative to proprietary systems. It handles the "thundering herd" of concurrent sessions through its asynchronous architecture and document-oriented storage. However, the documentation gap remains the primary barrier to wider FOSS adoption. With only 51 pages of official documentation and placeholders, users are forced to scavenge for implementation details on forums and wikis.

As for future trends, while TR-069 remains the industry standard for remote management systems, it is clear that it will soon be succeeded by more modern solutions such as the User Services Platform[16]. These modern frameworks aim to meet the needs of continuous communication and multiple controllers.

References

1. Zheng, L., Hu, Y., Chen, S.: Research and application of cwmp in distributed network management system. In: 2012 International Conference on Computer Science and Service System, pp. 647–650 (2012). https://doi.org/10.1109/CSSS.2012.167
2. Barczak, B., Kafel, T., Magliocca, P.: Network approaches and strategic management: exploration opportunities and new trends. J. Entrepreneurship Manage. Innov. **17**, 7–35 (2021). https://doi.org/10.7341/20211731
3. Rachidi, H., Karmouch, A.: A framework for self-configuring devices using TR-069. In: 2011 International Conference on Multimedia Computing and Systems (2011), pp. 1–6 (2011). https://api.semanticscholar.org/CorpusID:264024769
4. Stusek, M., Masek, P., Kovac, D., Ometov, A., Hosek, J., Kröpfl, F., Andreev 0001, S.: Remote management of intelligent devices: using TR-069 protocol in IoT. In: 39th International Conference on Telecommunications and Signal Processing, TSP 2016, Vienna, Austria, June 27–29, 2016, pp. 74–78. IEEE (2016). isbn: 978-1-5090-1288-6. https://doi.org/10.1109/TSP.2016.7760832
5. Stusek, M., et al.: A novel application of CWMP: an operatorgrade management platform for IoT. Int. J. Adv. Telecommun. Electrotechn. Signals Syst. **5**(3), 171–177 (2016)

[16] https://www.broadband-forum.org/pdfs/tr-369-1-3-0.pdf, accessed 8 May 2026.