

Advanced Out-of-Band management with BMCs and Redfish

Part 1. Bare Metal Foundations

This chapter delves into advanced aspects of Out-of-Band (OOB) management, focusing on Baseboard Management Controllers (BMCs) and the Redfish protocol. It explores key...

In this chapter

1. Introduction
2. Protocols and Systems in OOB Management
3. Challenges in Scaling OOB Management
4. Redfish Protocol and Vendor-Specific Complexities
5. Security in OOB Management
6. Performance and Scalability Considerations in OOB Management
7. Automation Strategies for OOB Management
8. Best Practices for Efficient OOB Management
9. Future Trends in OOB Management
10. Conclusion
11. Discussion Prompts

🕒 Chapter overview

Key Topics: Scaling and automation of OOB management, security and network isolation in OOB management, Redfish protocol and vendor-specific complexities

Learning Objective: By the end of this chapter, readers will be able to implement advanced Out-of-Band (OOB) management strategies using Baseboard Management Controllers (BMCs) and the Redfish protocol, while effectively addressing scalability, automation, and security challenges in diverse infrastructure environments.

Summary: This chapter delves into advanced aspects of Out-of-Band (OOB) management, focusing on Baseboard Management Controllers (BMCs) and the Redfish protocol. It explores key challenges such as scaling, automation, and security, providing strategies to efficiently manage diverse infrastructure environments. By implementing best practices, readers will gain a deeper understanding of how to enhance the reliability and security of their OOB management systems.

Introduction

As data centers expand, automation and remote management capabilities have become critical in maintaining operational uptime and stability. Out-of-band (OOB) management has emerged as a key technology for enabling administrators to manage systems regardless of the state of the operating system, power, or network conditions. By providing a dedicated, isolated control plane, OOB management ensures that essential maintenance tasks—such as system resets, firmware updates, and remote troubleshooting—can be performed without direct access to the system’s main operating environment.

This chapter explores the advanced aspects of OOB management, with a particular focus on Baseboard Management Controllers (BMCs) and the Redfish protocol.

While the concept of OOB management is straightforward, the real-world implementation introduces significant complexities. This chapter delves into these challenges and discusses strategies for overcoming them. We will cover best practices for scaling OOB management, integrating BMCs into larger automation systems, securing OOB interfaces, and dealing with vendor-specific nuances. By the end of this chapter, you will gain an in-depth understanding of how to effectively implement, manage, and scale OOB management systems in complex infrastructure environments.

Protocols and Systems in OOB Management

Out-of-Band (OOB) management relies on various vendor-specific and standardized protocols that allow administrators to manage and control systems remotely.

Vendor-Specific OOB Systems

iDRAC (Integrated Dell Remote Access Controller): Dell's iDRAC is a proprietary OOB management tool that allows remote monitoring and control of Dell servers, including access to consoles, power settings, and health data. It supports troubleshooting and updates via web interface, CLI, and APIs, without requiring physical access.

iLO (Integrated Lights-Out): HP's iLO provides similar remote management to iDRAC, allowing rebooting, firmware updates, and configuration management of HP servers through a secure interface. It supports access via web-based GUI or APIs for automation.

Other Vendor-Specific Systems: Vendors like Lenovo and Supermicro have their own OOB management systems (e.g., Lenovo's XClarity, Supermicro's IPMI-based solution). These systems have similar capabilities but differ in interfaces, creating challenges in multi-vendor environments.

IPMI (Intelligent Platform Management Interface)

IPMI is one of the earliest standards for OOB management, offering basic functionality such as remote power control, hardware health monitoring, and event logging. It allows administrators to manage servers without needing direct access to the operating system. One of the main advantages of IPMI is that it is a widely implemented standard across many vendors, making it a baseline protocol for OOB tasks.

While IPMI provides basic management functionality, it has limitations. One issue is that IPMI's security model is outdated, often relying on static passwords that can lead to vulnerabilities if not properly managed. IPMI also lacks extensibility and can be inconsistent across different vendors. In the transcript, the discussion emphasizes that while IPMI was an important development, it struggles to meet modern demands for security and functionality in large-scale, multi-vendor deployments.

Redfish

Redfish was developed as a modern, secure replacement for IPMI, using web protocols (HTTP, HTTPS) and formats (JSON) for more flexible remote management. It supports tasks like firmware updates, BIOS configurations, and network management.

Though designed to standardize management across vendors, Redfish implementations vary, especially between server generations and firmware versions. This inconsistency often leads to reliance on proprietary APIs or IPMI for older models. While Redfish improves upon IPMI, it doesn't fully solve the challenges of multi-vendor environments, requiring vendor-specific tools or scripts for full compatibility.

BMCs (Baseboard Management Controllers)

Baseboard Management Controllers (BMCs) are embedded systems essential for OOB management, handling tasks like power cycling, hardware monitoring, and console access through interfaces like IPMI, Redfish, or vendor-specific APIs (e.g., iDRAC, iLO).

Managing BMCs at scale presents challenges, such as default credentials, static IPs, and manual configuration, leading to complexity. Automation can help but doesn't fully eliminate these issues. Normalization—applying consistent configurations across systems—can streamline provisioning but faces hurdles, especially with legacy or mixed-vendor environments.

For more information on BMC normalization, plug these search terms into ChatGPT:

- "BMC normalization in OOB management"
- "Managing BMCs in mixed-vendor environments"
- "BMC configuration automation at scale"

Challenges in Scaling OOB Management

Scaling out-of-band (OOB) management introduces several key challenges that must be addressed to maintain efficiency, security, and operational consistency across a large infrastructure. These challenges arise as organizations attempt to apply OOB management practices, designed for small systems, to thousands of machines across diverse hardware platforms. Below, we explore the core challenges that surface when scaling OOB management.

Discovery and Configuration at Scale

One of the primary challenges when scaling OOB management is discovering and configuring new devices across a large infrastructure. In smaller environments, identifying a device, logging in, and manually configuring its BMC (Baseboard Management Controller) is relatively straightforward. However, at scale, this process becomes more complex due to the vast number of machines, each potentially having unique credentials, IP addresses, and default settings.

For example, many devices are shipped with randomized default passwords due to regulatory requirements (e.g., California laws mandating unique default passwords). While this improves security, it also complicates the process of automating configuration at scale. Organizations must have systems in place to securely gather these credentials, apply consistent configurations, and ensure that each device is properly integrated into their OOB management framework.

Credential Management and Access Control

Managing credentials across thousands of devices is another significant challenge in scaling OOB management. Ensuring that BMC credentials are secure, consistent, and properly rotated requires centralized control, but inconsistencies can arise when devices are added over time. This can lead to mismatches in access credentials, which complicates automation and security.

Further complicating this challenge is the fact that most organizations need to manage credentials in a way that doesn't impede automation workflows. Using centralized authentication systems like LDAP or Active Directory can help, but organizations must ensure that local accounts and fallback credentials are securely managed, stored, and periodically rotated.

Vendor and Hardware Generation Variability

A significant challenge in scaling OOB management is the variability across vendors and hardware generations. Each vendor typically implements their own BMC protocols, such as Dell's iDRAC, HP's iLO, and Lenovo's XClarity. Even though standards like Redfish have emerged to unify BMC management across vendors, inconsistencies remain in how each vendor implements these protocols. This makes it difficult to create a fully standardized automation process for managing BMCs across a large infrastructure with diverse hardware.

Moreover, within a single vendor's product line, different hardware generations may introduce new features, performance configurations, or BIOS settings that vary from previous models. This requires management systems to account for these differences and handle them effectively, which adds significant complexity to automation scripts and operational processes.

Performance Bottlenecks in Management Tools

OOB management interfaces are often constrained by the limited processing power of the BMCs themselves. When scaling management operations, such as firmware updates or configuration changes, these interfaces can become bottlenecks, particularly if the management tool only allows a limited number of concurrent actions. For example, updating firmware across thousands of machines might take significant time if the system is only able to handle a small batch of operations at a time.

Vendor-specific tools often exacerbate this issue, as they are typically designed to manage only a subset of machines concurrently. In large infrastructures, this creates inefficiencies in applying updates or executing maintenance tasks. Offloading tasks to individual machines, rather than relying on a central management system, can alleviate some of these performance bottlenecks, but it also adds complexity to the management process.

Security and Network Segmentation

As OOB management scales, securing these management interfaces becomes increasingly difficult. BMCs often operate on separate management networks, but ensuring that these networks remain properly segmented from production environments is critical for preventing unauthorized access. Using shared network ports for both production traffic and OOB management introduces security risks, as attackers could potentially gain access to BMCs through the production network.

Best practices for scaling OOB management include using dedicated, isolated networks for BMC traffic and ensuring strong authentication and encryption protocols are in place. Organizations also need to implement security monitoring systems that can track access to BMCs and alert administrators to potential breaches or unauthorized activity. However, the increased attack surface that comes with scaling OOB management requires a more sophisticated security posture, including stricter network controls and proactive monitoring of access patterns.

Redfish Protocol and Vendor-Specific Complexities

The Redfish protocol was introduced as a modern alternative to older management protocols like IPMI. It aims to standardize the management of hardware across vendors, offering a consistent interface for out-of-band management. However, despite Redfish's goal of unification, vendor-specific complexities still pose significant challenges in implementation, as manufacturers customize their implementations of Redfish and other protocols.

Overview of Redfish Protocol

Redfish is a RESTful API protocol designed to simplify server management across different hardware platforms. It allows administrators to interact with out-of-band management controllers like BMCs over HTTP(S), using JSON data formats for requests and responses. Redfish provides a standardized way to manage server hardware, including the ability to:

- Monitor system health (e.g., CPU, memory, power status)
- Control power and boot settings
- Manage firmware updates
- Configure BIOS settings
- Retrieve system inventory and hardware details

One of Redfish's key strengths is its use of well-defined endpoints and resources, which allows for consistent interaction with hardware management functions. This provides a more modern, secure, and flexible alternative to older protocols like IPMI.

Vendor-Specific Implementations

Despite Redfish's intention to create a unified standard, each hardware vendor tends to implement the protocol differently. Vendors often customize their Redfish implementations to support proprietary hardware features or differentiate themselves from competitors. As a result, while Redfish provides a common framework, its implementation can vary significantly from one vendor to another.

For instance, a Dell server's Redfish API may expose certain system performance metrics or BIOS settings that are not available on an HP server's Redfish interface. Additionally, some vendors may add proprietary extensions or custom endpoints that provide access to hardware-specific features not covered by the Redfish standard.

This variation can create challenges for administrators managing heterogeneous environments. In practice, despite the overarching standardization of Redfish, organizations must account for vendor-specific nuances in how Redfish APIs behave and what capabilities they expose. This requires a deep understanding of each vendor's Redfish implementation to ensure effective integration and management.

Redfish API Challenges

One of the main challenges with Redfish is the inconsistencies between vendors' API implementations. While Redfish defines a common schema, vendors can deviate from the standard or only partially implement the specifications. These deviations create compatibility issues and require custom handling for each vendor's hardware.

- **Incomplete Implementation:** Some vendors may implement only a subset of the Redfish specification, meaning certain Redfish endpoints or features may be unavailable or behave differently on different hardware.
- **Version Discrepancies:** Redfish implementations often differ between hardware generations. A system running an older version of Redfish may lack features or use different schemas compared to newer models. For example, an older BMC firmware may not support newer Redfish features introduced in subsequent revisions.
- **Proprietary Extensions:** Vendors frequently introduce proprietary extensions that are specific to their hardware. These extensions are outside the Redfish standard but offer additional functionality. However, they also mean that generic tools built for Redfish may not be fully compatible across all vendors without custom development.

Redfish's Role in Multi-Vendor Environments

By using Redfish's standardized API structure, administrators can develop tools and scripts that work across different hardware, reducing the need to use vendor-specific management software for each type of server.

That said, multi-vendor environments still require careful handling of the idiosyncrasies introduced by each vendor. Automated management tools must be designed to detect and adapt to the specific Redfish implementation of the target hardware. In many cases, tools will need to query the capabilities and available endpoints of each Redfish interface before executing actions, ensuring that the hardware supports the desired operations.

Organizations that rely on Redfish for multi-vendor environments need to maintain an updated knowledge base of vendor-specific behaviors and differences, including any custom extensions or variations in implementation. This ensures that their management tools remain compatible across all hardware, reducing the likelihood of failed operations or unintentional misconfigurations.

Integration with Vendor Tools

While Redfish is designed as a universal standard, many organizations still rely on vendor-specific tools to manage their hardware. Tools like Dell's iDRAC, HP's iLO, and Lenovo's XClarity are often more mature and fully featured than their Redfish counterparts, as they are tailored to the specific vendor's hardware and capabilities.

For example, vendor tools often include optimized workflows for tasks such as bulk firmware updates, automated BIOS configuration, or detailed hardware diagnostics. These tools may also offer better support for advanced features like power management, hardware performance tuning, or security configurations. Even if Redfish supports these features in theory, vendor tools may provide a more reliable or faster implementation.

To balance standardization with the power of vendor tools, many organizations use a hybrid approach: leveraging Redfish for basic cross-vendor operations while relying on vendor-specific tools for more complex tasks. Automation systems can be designed to switch between Redfish and vendor tools depending on the required action, ensuring that they use the most efficient and reliable method available.

Best Practices for Redfish and Vendor-Specific Management

Given the challenges of Redfish and vendor-specific implementations, there are several best practices organizations can follow to ensure a smooth OOB management experience:

- When deploying new hardware or upgrading firmware, always test the Redfish API to ensure that it provides the necessary capabilities. Be prepared to fall back on vendor-specific tools if Redfish lacks support for a required feature.
- Keep detailed documentation of each vendor's Redfish implementation, including supported endpoints, custom extensions, and known quirks. This documentation will help inform automation development and troubleshooting.
- For basic tasks like power management, inventory retrieval, and hardware health monitoring, Redfish can be a good fit. These operations tend to be well-supported across vendors and are less prone to vendor-specific differences.
- For more advanced or hardware-specific tasks—such as tuning BIOS settings, managing storage configurations, or applying bulk firmware updates—use the vendor's native management tools. These tools are typically more robust and better suited to the vendor's unique hardware.
- Automation tools should be designed to detect and adapt to the specific Redfish version and vendor extensions in use on each machine. This ensures that automation processes work consistently, even across different hardware models or firmware versions.

To learn more about Redfish protocol variations across vendors, plug these search terms into ChatGPT:

- "Redfish API vendor differences"
- "Redfish vs IPMI comparison"
- "Redfish protocol implementation challenges"

Security in OOB Management

Poorly secured OOB management interfaces can become gateways for attackers, granting them access to critical infrastructure. Securing these interfaces requires a combination of network isolation, authentication controls, encryption, and continuous monitoring to ensure that unauthorized access is prevented.

Network Isolation and Segmentation

The most fundamental step in securing OOB management interfaces is to isolate them from the production network. This isolation ensures that even if the production network is compromised, attackers cannot directly access the management controllers. Assigning a dedicated network specifically for OOB management traffic reduces the attack surface. The network should be limited to only essential systems, such as the management servers and trusted administrators. It should also be protected by firewalls that limit the flow of traffic into and out of the network. Virtual LANs (VLANs) can be used to logically separate OOB management traffic from other network traffic, helping to protect the interfaces from lateral movement within the network in case of a breach.

Authentication and Access Control

Strong authentication and access control policies are essential to securing OOB management systems. Access should be restricted to authorized users only, following the principle of least privilege. Implementing RBAC allows organizations to enforce granular access controls based on user roles. Whenever possible, multi-factor authentication should be used to secure access to OOB management interfaces. Enforce strong password policies for local accounts on the BMC (Baseboard Management Controller) and other OOB management interfaces. Default credentials must always be replaced with unique, complex passwords for each system.

Encryption and Secure Communication

The use of secure communication protocols, such as HTTPS and TLS, ensures that data in transit between the management console and the OOB interfaces is encrypted and protected from eavesdropping or man-in-the-middle attacks. Organizations should implement proper certificate management practices to validate and secure connections between management servers and OOB interfaces. Self-signed certificates should be avoided, and certificates issued by trusted Certificate Authorities (CAs) should be used to ensure authenticity.

Continuous Monitoring and Alerting

OOB management interfaces should be actively monitored for signs of unauthorized access or misuse. Logging access attempts, successful logins, and configuration changes on OOB management systems enables security teams to detect unusual or suspicious activities. For example, frequent failed login attempts may indicate a brute-force attack.

Many OOB management systems, particularly those from enterprise vendors like Dell, HP, and Lenovo, support Simple Network Management Protocol (SNMP) and syslog. These features can be integrated with centralized monitoring systems to collect and analyze data in real time. Alerts can be configured to notify administrators if unusual activity is detected, such as a login from an unexpected IP address or a configuration change outside of normal maintenance windows.

As an added security measure, intrusion detection systems (IDS) and intrusion prevention systems (IPS) can be deployed to monitor the OOB management network. These systems detect and block malicious traffic aimed at OOB interfaces.

Best Practices for Securing BMCs

- Any services that are not actively used should be disabled to reduce the attack surface.
- Keeping BMC firmware up to date is crucial for maintaining security. Firmware updates often include patches for vulnerabilities, and failing to apply them can leave OOB management interfaces exposed to known exploits.
- Maintain a local emergency access account on the BMC with a unique, complex password. This account can serve as a backup if the centralized authentication system is unavailable. Access to this account should be tightly controlled and regularly audited.

Air-Gapped and High-Security Environments

In air-gapped environments, the OOB management network should be completely isolated from external networks, including the internet and the primary production network. Physical security measures, such as securing access to the management network's hardware, become even more critical in these setups.

In environments where BMCs are used but network access is limited, ensure that all access is conducted via secure, controlled endpoints. Only trusted devices should be allowed to connect to the management network, and these devices should undergo regular security audits.

Enabling event-driven alerts for BMC access and monitoring is essential in high-security environments. For example, an alert can be triggered when an administrator logs into a BMC, allowing security teams to verify that the access is authorized.

For more information on credential and access management, plug these search terms into ChatGPT:

- "Scaling credential management in OOB"
- "OOB management access control"
- "Automating credential rotation for BMCs"

Performance and Scalability Considerations in OOB Management

Out-of-band (OOB) management becomes increasingly complex when applied at scale, particularly in large data centers or enterprise environments where hundreds or thousands of servers must be managed efficiently. Performance and scalability are key challenges that must be addressed to ensure that OOB management can handle the growing demands of modern infrastructure.

Bottlenecks in Centralized Control

One of the primary challenges in scaling OOB management is the potential bottleneck introduced by centralized control mechanisms. Many OOB systems rely on a central server or management tool to interface with the BMCs (Baseboard Management Controllers) of individual machines. This centralized server is responsible for tasks like updating firmware, applying configuration changes, and monitoring hardware health. At a small scale, this approach works well, but as the number of managed devices grows, performance issues can emerge:

- **Limited Concurrency:** Many vendor tools, especially those that rely on older protocols like IPMI, have limited concurrency, meaning they can only manage a fixed number of devices simultaneously. For instance, an HP management platform might be limited to managing 16 devices at a time, even if the server manages over 700 machines.
- **High Latency:** Interactions with BMCs are often slower than traditional network operations due to the limited processing power of BMCs. For example, even with modern Redfish-based management, some actions—such as applying configuration changes or updates—require significant dwell time for completion.
- **Vendor-Specific Constraints:** Certain vendor tools impose restrictions on how many devices can be managed concurrently. This limitation can significantly delay operations like firmware updates or BIOS changes across an entire fleet of servers, slowing the overall process.

Distributed Workloads to Improve Efficiency

A solution to centralized bottlenecks is to distribute the workload across the managed devices themselves, rather than running all operations from the central management server. Instead of driving all changes through the OOB interface, in-band operations—executed within the operating system itself—can be used to push configuration changes, firmware updates, or other management tasks. Once the tasks are completed in-band, they can update the BMC to ensure the OOB management system remains in sync.

By distributing management tasks to be executed in parallel across the machines themselves, large-scale updates can be completed faster. For example, instead of pushing a firmware update to 16 machines at a time from a central server, each machine could pull the update and apply it independently, resulting in significantly faster rollouts.

Balancing In-Band and Out-of-Band Management

While in-band management can enhance scalability, it's important to balance in-band and OOB operations to ensure maximum reliability and performance:

Task Segmentation: Not all tasks need to be handled through OOB management. Firmware updates, configuration changes, and diagnostics that can be safely handled through the in-band system should be offloaded to the operating system, with only critical or low-level management functions (such as power cycling) handled via OOB.

Fallback to OOB: In situations where the operating system is unavailable or unresponsive, OOB management remains essential for recovery and maintenance. Thus, while in-band operations may improve efficiency, OOB must still serve as the fallback for critical management functions.

Scaling Through Automation

Automation is key to managing large fleets of servers with OOB management. Manual processes do not scale effectively, and human error increases as the number of devices grows. The following strategies can help improve scalability through automation:

- **Automated Discovery:** Automating the discovery of new devices and their BMCs ensures that newly racked or deployed servers are automatically integrated into the management platform. This reduces the manual workload and ensures consistent management practices across all devices.
- **Automated Updates:** Implementing automated workflows for BIOS updates, firmware patches, and configuration changes allows large numbers of servers to be managed simultaneously. These workflows can be designed to execute in parallel and monitored for failure to ensure success without human intervention.
- **Health Checks and Monitoring:** Automating health checks and monitoring through OOB management interfaces helps identify issues before they become critical, enabling proactive maintenance at scale.

Mitigating Resource Consumption

Another factor to consider in large-scale OOB management is resource consumption. The BMCs themselves often have limited processing power and memory, and some actions—such as logging, updating, or configuring—can overload these resources if executed on a large number of machines simultaneously.

Where possible, batch operations should be optimized to minimize resource consumption. For example, instead of setting BIOS parameters one at a time, a batch of changes should be applied in a single operation to reduce the number of interactions with the BMC.

Using newer, more efficient protocols like Redfish—when available—can improve performance and reduce the number of interactions needed to complete management tasks. Redfish's use of JSON and HTTP(S) allows for more streamlined communication compared to older protocols like IPMI.

For more information on performance bottlenecks in OOB management tools, plug these search terms into ChatGPT:

- "BMC performance limitations"
- "Scaling firmware updates with OOB tools"
- "Reducing latency in OOB management"

Automation Strategies for OOB Management

Automation is key to effectively managing large-scale infrastructures through out-of-band (OOB) management. Automating processes helps mitigate human error, enhance security, and increase efficiency.

Automated Discovery and Provisioning

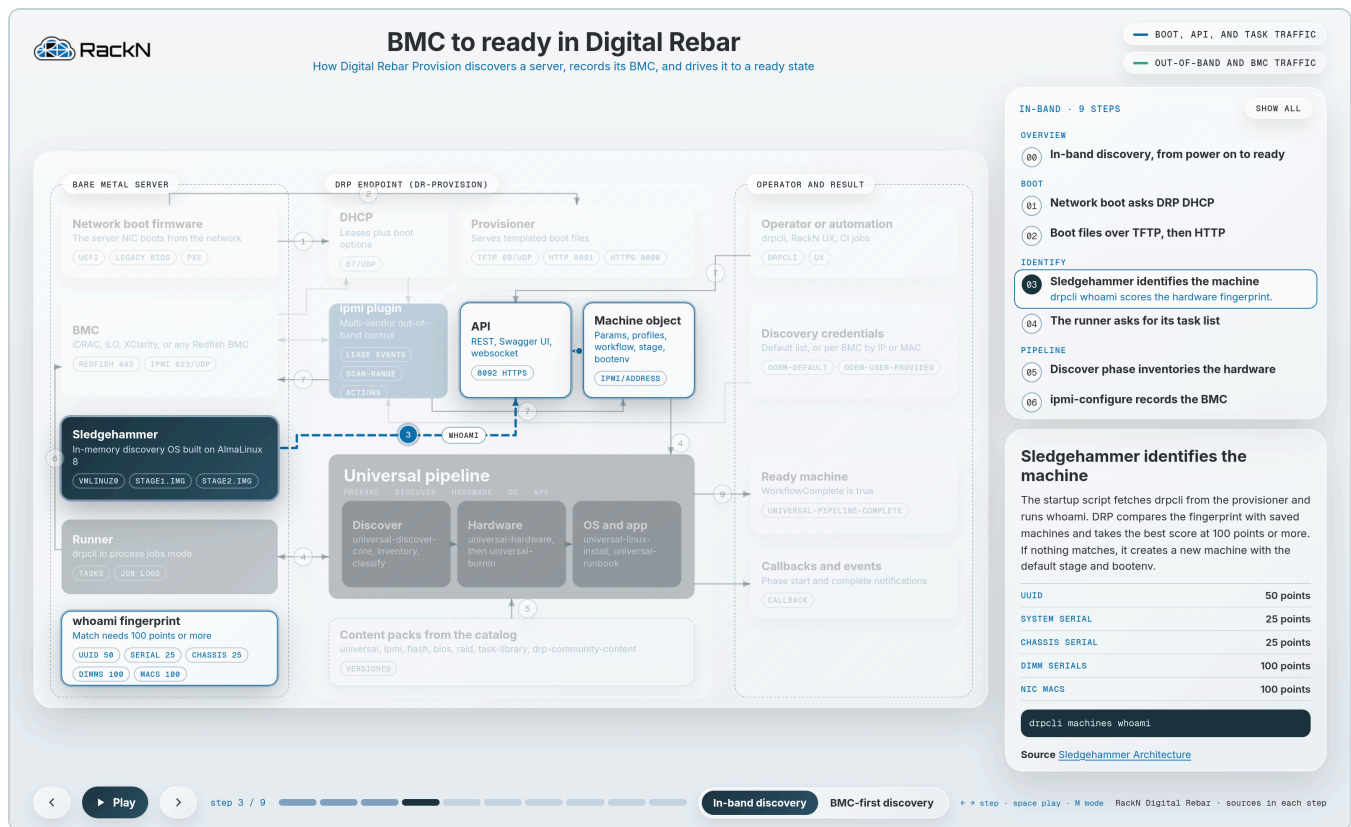
Automated discovery tools scan for and identify BMC devices on the network, automatically assigning IP addresses through DHCP and registering them into the system. This eliminates manual entry of device information and enables quicker onboarding. Credential automation can reset default passwords to corporate standards, standardizing security across all devices.

Interactive Walkthrough of BMC-to-Ready Onboarding

Added for this preview

This walkthrough and the reference below are new. They are not part of the original chapter text. Every step cites the public Digital Rebar documentation it comes from, and steps the docs do not settle are marked unverified.

The walkthrough follows one server through Digital Rebar Provision (DRP) in two ways. **In-band discovery** starts with a network boot into Sledgehammer. **BMC-first discovery** starts with the BMC taking a DHCP lease, which matches the brownfield and scale problems this chapter describes. Use the arrows, click a step, or press play.



Step 3 of the in-band path in the interactive walkthrough. The online chapter has the full, clickable version with both discovery paths and a source link on every step.

Digital Rebar reference for this section

The commands below come from the RackN IPMI plugin documentation. Check them against the docs for your DRP version before you run them.

```
# Install the ipmi plugin from the catalog
drpcil catalog item install ipmi

# Find BMCs in an address range over Redfish (256 addresses per scan at most).
# Needs ipmi/discover/enabled=true on the plugin and discovery credentials.
drpcil plugins runaction ipmi scan-range \
  ipmi/discover/scan-start-address "10.0.0.10" \
  ipmi/discover/scan-end-address "10.0.0.50"

# List scan results, then show one by its ID
drpcil extended -l ipmi_scan_results list
drpcil extended -l ipmi_scan_results show <scan-id>

# Out-of-band actions on a machine once ipmi/address, ipmi/username,
# and ipmi/password are set
drpcil machines runaction <machine-uuid> nextbootpxe
drpcil machines runaction <machine-uuid> powercycle
```

```
# The same actions through the API (HTTPS API port 8092 by default)
curl -k -u username:password -X POST \
  https://drp-endpoint:8092/api/v3/machines/<machine-uuid>/actions/powercycle
```

Plugin parameter	What it does	Default
<code>ipmi/discover/enabled</code>	Turns on OOBM discovery in the plugin	not stated
<code>ipmi/discover/from-lease</code>	Runs discovery when DRP DHCP leases an address to an unknown system	not stated
<code>ipmi/discover/oobm-default-credentials</code>	Credentials tried on every controller	set by you
<code>ipmi/discover/oobm-user-provided-credentials</code>	Credentials for one controller, keyed by IP or MAC	set by you
<code>ipmi/discover/parallel</code>	Lease events processed in parallel	200
<code>ipmi/discover/buffer-size</code>	Queued events before new ones are dropped	20000
<code>ipmi/discover/workflow</code>	Workflow set on each new machine	<code>universal-discover</code>
<code>ipmi/mode</code>	Toolset for actions (<code>ipmitool</code> , <code>racadm</code> , <code>redfish</code> , <code>lpar</code>)	<code>ipmitool</code>

Sources are the [IPMI Plugin Usage](#) page, the [Multi-API OOB/BMC Control](#) developer page, and the [ipmi/mode](#) reference. Autodiscovery and range scan support Redfish only today, according to the plugin docs.

Configuration and Firmware Automation

Automation enables firmware updates to be pushed in parallel across a fleet of devices, drastically reducing downtime and administrative overhead. Configuration templates can be applied to BMCs to standardize settings such as network configurations, minimizing the risk of errors due to manual input. This ensures consistency and simplifies operations management.

Error Handling and Recovery

Automation can include regular health checks and self-healing processes, allowing devices to recover automatically from failures. Automated recovery systems revert configurations to a stable state when errors are detected during updates or changes, ensuring system resilience.

Parallelization and Task Distribution

By distributing tasks like updates or reboots across multiple devices in parallel, automation can significantly speed up processes and reduce downtime. This avoids bottlenecks and ensures that the workload is balanced across the infrastructure, making it more scalable and efficient.

CI/CD Integration

Automation can be integrated with Continuous Integration and Continuous Deployment (CI/CD) pipelines, allowing synchronized hardware and software updates. This ensures that infrastructure management remains agile and keeps up with rapid deployment cycles.

Security Automation

Automated security features can rotate credentials, update certificates, and manage access control configurations across BMCs, ensuring compliance with security policies without manual intervention. This protects sensitive management interfaces from unauthorized access and minimizes risk.

For more information on automation in OOB management, plug these search terms into ChatGPT:

- "OOB automation strategies"
- "Automating BMC discovery and provisioning"
- "Automating health checks and recovery in OOB systems"

Best Practices for Efficient OOB Management

Efficient out-of-band (OOB) management is critical for maintaining the security, scalability, and reliability of large infrastructure environments. Adopting the following best practices can ensure that OOB management systems are well-optimized and resilient.

Standardize Configurations

Establish standardized configurations for all OOB devices. Use configuration templates and automation to ensure consistency across devices, avoiding configuration drift and reducing potential misconfigurations. Ensuring that all devices adhere to a common baseline simplifies management and troubleshooting.

Segregate OOB Networks

OOB management networks should be physically or logically separated from production networks. This segregation limits the risk of unauthorized access and ensures that management traffic does not interfere with production workloads. Use VLANs or physically isolated switches to create secure, dedicated pathways for OOB traffic.

Automate Security Management

Enforce strict security protocols for OOB management interfaces by automating password updates, certificate management, and access control. Utilize LDAP or Active Directory integration to centralize authentication, and configure BMCs to use role-based access controls (RBAC) to ensure that only authorized personnel have access to sensitive management functions.

Monitor and Audit OOB Access

Enable logging and monitoring on all OOB devices to capture access and configuration changes. Centralize logs using Syslog or other log aggregation tools, and set up alerts for suspicious activity or unauthorized access attempts. Regular audits of OOB access can help identify potential security issues before they escalate.

Implement Redundancy

Ensure that OOB management systems are highly available by implementing redundancy at the network and device level. Have backup OOB interfaces available in case of failure and consider dual-homed BMC configurations where possible. This ensures that even in the event of an outage, you maintain access to critical management functions.

Leverage Automated Health Checks

Set up automated health checks for OOB systems to ensure that BMCs and other management interfaces remain functional. Automate alerts and remediation actions for when issues are detected, reducing downtime and the need for manual intervention. Automated checks can ensure that systems are responsive and healthy at all times.

Prioritize Firmware Management

Regularly update firmware on BMCs and other management components. Outdated firmware can have security vulnerabilities or compatibility issues that could jeopardize system stability. Automate firmware updates and rollbacks where possible, ensuring all devices are consistently patched.

Use Scalable Solutions

Choose OOB management tools and systems that scale effectively with your infrastructure. Ensure that the management platform supports parallel execution and can handle the increased workload as your environment grows. Plan for scalability from the beginning to avoid bottlenecks as the number of managed devices increases.

Test and Validate Changes

Before deploying changes to OOB systems, whether in configuration, firmware, or automation workflows, thoroughly test them in a lab or staging environment. Validate the changes to ensure they won't introduce downtime or other unintended consequences when rolled out to production systems.

Document Everything

Maintain clear and up-to-date documentation for all aspects of OOB management, including configurations, security protocols, and automation workflows. This ensures that team members can quickly understand and manage OOB systems and provides a clear audit trail for compliance and troubleshooting.

Future Trends in OOB Management

As technology advances and infrastructure complexity grows, out-of-band (OOB) management continues to evolve. The following trends highlight how OOB management is adapting to meet the demands of modern computing environments.

Enhanced Security with Zero Trust Architectures

Security remains a top priority for OOB management, especially as cyber threats continue to evolve. The shift towards zero trust architectures is influencing OOB strategies, where each access request, even within the network, is verified before being granted. This approach significantly reduces attack surfaces by implementing stricter identity verification, multi-factor authentication (MFA), and role-based access controls (RBAC) in OOB systems. Expect future OOB systems to fully integrate zero trust principles to further secure management interfaces.

Expansion of Redfish Adoption and Standards

The Redfish protocol is poised to see even broader adoption as it matures and evolves. Redfish's standardized API for managing and monitoring hardware across vendors simplifies cross-platform management. As more vendors adopt and extend Redfish functionality, OOB management will become increasingly vendor-agnostic, allowing for greater interoperability and standardization in large, diverse environments.

Open Source OOB Management Platforms

The future will likely see increased adoption of open-source OOB management platforms, driven by a need for greater flexibility, cost control, and community collaboration. Open-source solutions allow for greater customization and adaptability to unique infrastructure needs. These platforms, such as OpenBMC, are gaining traction as organizations look to reduce vendor lock-in and build more transparent and customizable management systems.

Energy Efficiency and Sustainability Initiatives

As sustainability becomes a top priority for data centers and IT operations, OOB management will evolve to include energy efficiency metrics and controls. Future OOB solutions may include advanced power management features that allow for real-time monitoring of energy usage, automated shutdown of idle systems, and optimizations for reducing power consumption. These tools will play a crucial role in supporting green IT initiatives and achieving sustainability goals.

Conclusion

Out-of-band (OOB) management is a critical component of modern IT infrastructure, providing essential control and visibility over systems when traditional methods may not be available. As explored in this chapter, mastering OOB management requires understanding the core protocols and systems, addressing challenges related to scaling, ensuring robust security, and implementing effective automation strategies.

From the integration of the Redfish protocol to vendor-specific complexities, the importance of lifecycle management and automation cannot be overstated. Managing devices at scale demands efficiency, consistency, and strategic approaches to avoid downtime and security risks. By employing best practices such as isolating OOB networks, utilizing automation for updates and monitoring, and establishing secure credentials, organizations can achieve more reliable and scalable OOB management.

By understanding and applying the principles outlined in this chapter, software engineers and system administrators can build robust OOB management frameworks that enhance the resilience and security of their infrastructures, positioning them for success in a rapidly evolving technological landscape.

Discussion Prompts

1. Redfish Protocol Implementation

1. How does Redfish improve upon IPMI, and what are the challenges that still exist with vendor-specific implementations? What are potential solutions to these inconsistencies across different hardware platforms?

2. Scaling OOB Management

1. What are the biggest challenges when scaling out-of-band management across large, heterogeneous environments, and how can automation help address these issues? How can organizations effectively manage credential and network configurations at scale?

3. BMC Normalization Strategies

1. What are the benefits and challenges of normalizing BMC configurations across a mixed-vendor environment? How can automation be leveraged to enforce consistent configurations across legacy and modern systems?

4. Security in OOB Management

1. What are the best practices for securing OOB management networks and interfaces, and how can organizations balance security with accessibility? How does the shift towards zero trust architecture impact OOB management strategies?

5. Performance Bottlenecks and Optimization

1. How do the processing limitations of BMCs affect the performance of OOB management at scale? What strategies can be implemented to minimize bottlenecks and improve the efficiency of tasks like firmware updates?

6. Automation in OOB Management

1. In what ways can automation streamline the discovery, provisioning, and maintenance of devices managed via OOB interfaces? How can organizations design automation systems that effectively balance in-band and out-of-band operations?

7. Vendor-Specific Tools vs. Redfish

1. How should organizations decide when to rely on Redfish versus vendor-specific tools like iDRAC or iLO? What factors should be considered when integrating both approaches into an automation pipeline?

8. Future Trends in OOB Management

1. How will advancements in areas like cloud-native infrastructure, edge computing, and zero trust security reshape the future of OOB management? What new challenges and opportunities might arise as OOB management evolves to meet the needs of these emerging technologies?