Macrium Software

INDUSTRY GUIDE

Designing Recovery and Resilience for Embedded Medical Devices

How OEMs can make device recovery a **built-in capability**, not an afterthought.

Contents

Medical devices **are no longer just hardware.**

They are software-defined systems. As embedded operating systems become central to diagnostics, imaging, and clinical automation, device availability is now determined by software reliability, not mechanical durability.

When those systems fail (due to corruption, failed updates, or hardware issues) the entire device can become unusable. In clinical environments, that means disrupted workflows, delayed diagnoses, and rising service costs.

This guide outlines how medical device manufacturers can design recovery as a built-in system capability, enabling devices to return to a validated operational state quickly, consistently, and without complex service intervention.

With downtime costing hospitals an average of **£7,500 per minute**, recovery speed is no longer a support concern. **It is a core product requirement.**

The Hidden Risk: **Embedded System Failure**

Modern medical devices rely on embedded software to operate analysers, imaging platforms, and clinical automation tools. This fundamentally changes how failure occurs.

Devices no longer fail primarily because of hardware. They fail when the software fails: when the OS becomes corrupted, an update introdtuces incompatibility, or storage degrades. Over long device lifecycles, these failures are not rare. **They are inevitable.**

A medical device doesn't fail when hardware breaks. **It fails when the system won't boot.**

The Real Cost of **Device Downtime**

When a clinical device goes down, the impact is immediate. Diagnostic workflows stop, lab samples begin to queue, imaging schedules fall behind.

In many environments, downtime tolerance is measured in minutes:

Emergency care:
under 5 minutes

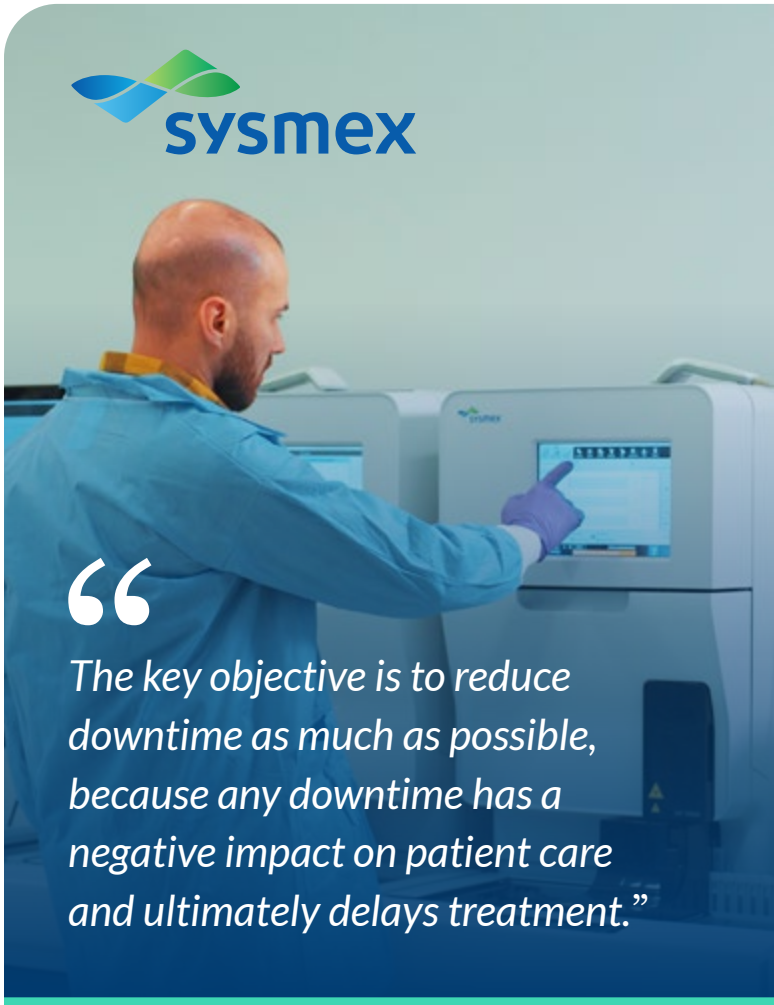
Lab diagnostics:
under 15 minutes

Recovery, however, often takes far longer, especially when it depends on engineer dispatch and manual rebuilds. The costs compound quickly:

£7,500
per minute of downtime

£150–£500
per hour for engineering support

SLA penalties
and reputational damage



"The key objective is to reduce downtime as much as possible, because any downtime has a negative impact on patient care and ultimately delays treatment."

MTTR is the most important metric OEMs don't design for.

Why Recovery Is **Harder Than It Looks**

In theory, device recovery is simple: restore the system and return to operation. In practice, it is far more complex. Four structural realities make fast, consistent recovery difficult to achieve without a purpose-built approach.

01 Air-Gapped Clinical Networks

Many clinical environments operate on restricted or isolated networks to protect sensitive patient data. **Recovery must work entirely locally**, with no reliance on cloud services, external validation, or downloads.

- Can recovery run without internet access?
- Does it depend on external licensing or activation?

02 Long Device Lifecycles

Medical devices are designed to remain in service for 7–12 years or more. Over that time, operating systems age, drivers fall out of sync, and hardware components change. **The longer a system is deployed, the greater the likelihood of failure.** If recovery was not designed for longevity from the start, restoring a device becomes significantly harder.

Your recovery strategy **must outlive your OS.**

03 Regulatory & Validation Constraints

Medical device software is governed by frameworks including IEC 62304 and ISO 13485. **Any system change can trigger revalidation,** an expensive and time-consuming process. The key distinction: restoring a system to a validated "known-good" state is considered servicing, not redesign. This makes **recovery architecture not just operationally important, but strategically valuable** from a compliance perspective.

04 Fleet Fragmentation

OEMs rarely support a single device configuration. Instead, they manage fleets spanning multiple hardware generations, operating systems, and regional variants. This creates **significant operational complexity** when recovery processes differ across devices.

- How many system images do you currently maintain?
- Can a single recovery process work across your entire fleet?

From Backup to **Device Resilience**

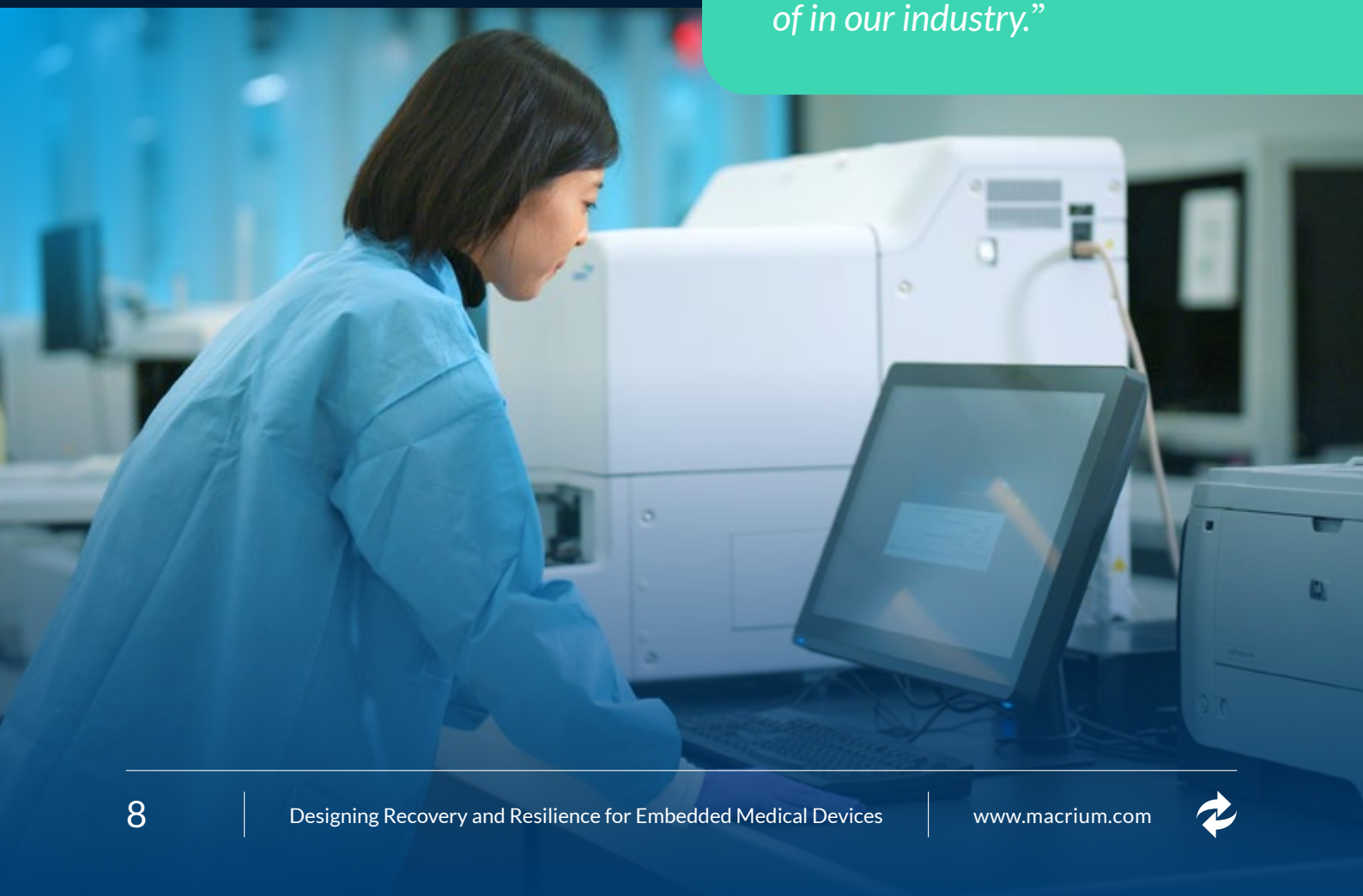
Backup is not the objective. Availability is.

In enterprise systems, protecting data makes sense, because data is the primary asset. In medical devices, the situation is different. Critical patient data is typically not stored on the device itself. It is transmitted to and managed by external systems (LIS, PACS, electronic patient records) where it is backed up as part of broader hospital infrastructure.

The device is responsible for processing, analysis, and operation. When a device fails, the problem is not lost data. It is lost functionality. Recovery means restoring the system to a working state, quickly and reliably.

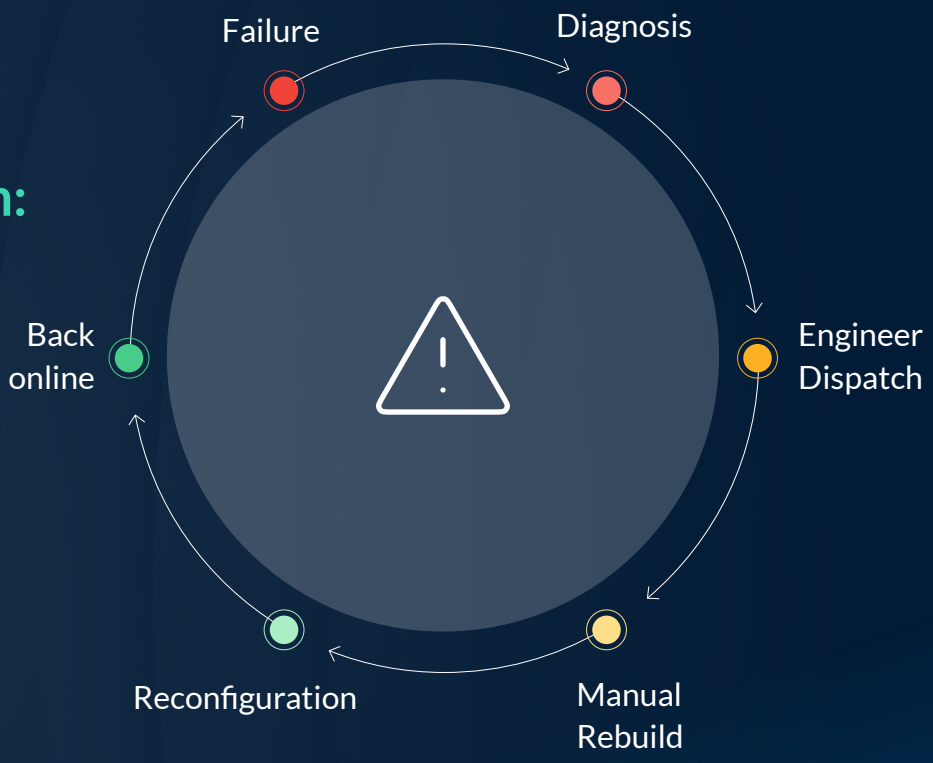
sysmex

“Being able to recover software issues remotely is truly unheard of in our industry.”

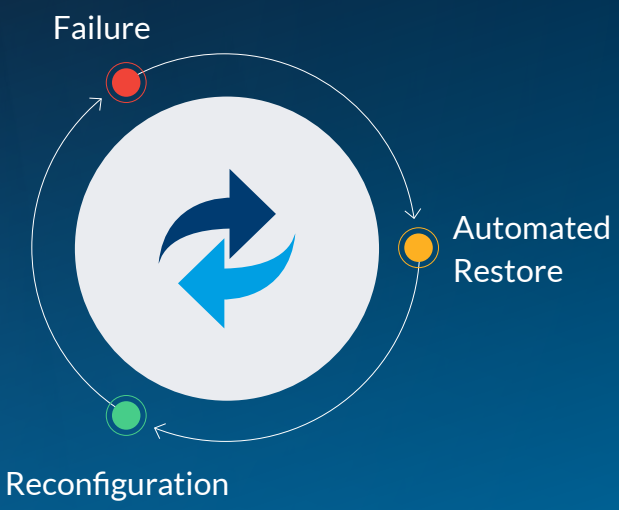


Reducing MTTR **Across Device Fleets**

Most recovery processes today follow a familiar and slow pattern:



This leads to recovery times measured in hours or days. A resilient approach replaces it with:



Traditional Recovery		Resilient Recovery	
1	Failure detected	1	Failure detected
2	Diagnosis attempt	2	Automated restore initiated
3	Escalation (TAC → FSE)	3	Device operational
4	Engineer dispatched		
5	Manual rebuild		
6	Reconfiguration & testing		
7	Device operational		

24–48 hours (or longer)

Multiple manual steps, dependent on engineer availability

10–20 minutes

Standardised, repeatable. No on-site engineer required.

sysmex

"Now our TAC agents can remotely restore systems and get them back up and running in a matter of minutes."

Designing Recovery into **Your Device Platform**

This section moves from problem to implementation. Recovery should be treated as a core system capability, designed in from the start, not bolted on after the fact.

The decisions made at design time determine whether devices can be restored in minutes or require hours of manual intervention.

01

Define Your Recovery Architecture

Every device platform needs a clearly defined recovery strategy before anything else. This is the foundation for everything that follows. At minimum, your architecture must answer:

- Where does recovery run from?
- What triggers it?
- What is restored, and how?
- What is the authoritative source for system state?

What good looks like

- Local-first, with no dependency on network or cloud
- Automated and repeatable
- Restores a validated, working system state
- Works consistently across all deployed device variants

Common mistake

Treating recovery as a rebuild process. Rebuilds introduce variability and delay. Recovery must be deterministic: **the same inputs must always produce the same outcome.**

02

Define Your Known-Good State

Recovery only works if you know exactly what you are restoring to. For some devices this is a golden image; for others it may be the last validated system state. Either way, it must be explicit, version-controlled, and traceable. Treat it like a software release, aligned with validated builds and tested as part of your QA process.

What good looks like

- A clearly defined, validated system state
- Includes OS, applications, drivers, and configuration
- Version-controlled, traceable, and QA-tested
- Consistent between what is shipped and what is recoverable

Common mistake

State drift: devices in the field no longer match any validated recovery point. This is the single most common reason recovery fails in practice.

03

Design for Speed and Repeatability

The primary objective of recovery design is to reduce MTTR. That means eliminating dependency on manual rebuild processes, specialist engineers, and complex troubleshooting. The target: any device can be restored to operation in minutes, by a non-engineer, with minimal steps (whether triggered locally or remotely by a TAC agent).

What good looks like

- Minimal steps to initiate recovery
- Fully automated restore process
- No requirement for deep technical expertise
- Supports local, remote-triggered, and escalation pathways

Common mistake

Designing recovery workflows that assume technical knowledge, manual intervention, or variability in execution. If a non-engineer cannot follow the process, it will fail under pressure.

04

Design for Offline, Real-World Environments

Medical devices do not operate in ideal conditions. Recovery must assume no internet connectivity, restricted environments, and limited access to external systems. If recovery depends on a cloud connection or external activation that is unavailable at the moment of failure, it is not a recovery system. It is a liability.

What good looks like

- No cloud dependency
- No external activation requirements
- All recovery assets available locally on the device

Common mistake

Building recovery processes that silently fail when network access is unavailable. Test recovery in completely isolated conditions. That is the environment that matters most.

05

Align Recovery with Device Lifecycle and Licensing

Devices remain in the field for a decade or more. Recovery must remain reliable for that entire period. The most easily overlooked risk: a licensing or subscription dependency that cannot be satisfied at the moment a device needs to recover. When that happens, recovery fails. Not because of a technical fault, but because of a commercial one.

What good looks like

- No dependency on active subscriptions or recurring license validation
- No ongoing connectivity required to enable recovery
- Recovery architecture tested against end-of-support scenarios

Common mistake

Treating licensing as a commercial decision separate from product design. If recovery depends on an active licence that may lapse during the device's operational life, the device is not truly resilient.

Macrium Reflect LTSC: **Recovery Built for Medical OEMs**

Design Requirement	How Reflect LTSC Delivers It
Offline and air-gapped operation	Operates fully without internet access. No cloud dependency, no external activation. Suitable for restricted and secure clinical network environments.
Long-term lifecycle stability	A static feature set with no UI changes or feature updates. Updates are limited to security patches and critical bug fixes, reducing the risk of revalidation triggered by unexpected changes.
Known-good system state	Full disk imaging captures the complete OS, settings, and applications. Rapid Delta Restore (RDR) restores only changed data blocks, reducing recovery time and minimising downtime.
No specialist required	Interactive and command line tools support installation and recovery across large device volumes, without requiring deep technical expertise on-site.
Lifecycle-stable licensing	One-time perpetual licensing with offline activation. No recurring subscription, no connectivity required to maintain licence validity across a device's full operational life.

"The competitive edge that Macrium gives us is that now we're able to recover software or image issues remotely, which is truly unheard of in our industry. We were able to save approximately 50% in labour related to software reimaging issues."

Jose Rivera, VP of Service and Support, Sysmex America

To find out how Macrium Reflect LTSC can be integrated into your device platform, contact us at **sales@macrium.com** or visit **www.macrium.com**

Building a Resilient **Device Platform**

A resilient platform is not built from any single decision. It emerges from the alignment of architecture, validated system state, operational model, and lifecycle planning.

When these elements work together, recovery becomes fast, consistent, and scalable across your entire fleet.

Implementation Roadmap

01 Audit current failure modes and service data to understand where time is lost today

02 Define and validate your known-good system images across all deployed configurations

03 Standardise your recovery architecture: local-first, offline-capable, no specialist required

04 Pilot the approach across one product line, measure MTTR, and scale across the fleet

Final Checklist

- ✓ Can your device recover to operation in under 20 minutes?
- ✓ Can recovery run fully offline, with no external dependencies?
- ✓ Is recovery tied to a validated, version-controlled system image?
- ✓ Can a non-engineer initiate and complete recovery without escalation?
- ✓ Is recovery consistent and repeatable across all device variants in your fleet?
- ✓ Is your licensing model stable and non-blocking for the full device lifecycle?

Recovery is not a feature you add later.

It is the measure of whether your device truly works, when it matters most.

Enable faster recovery, lower downtime, and reduced support costs with

Macrium Reflect LTSC

Learn more at

www.macrium.com

or email

sales@macrium.com

Macrium Software