



Memfault

**Boosting IoT Product
Performance and Quality
with Device Reliability
Engineering**

François Baldassari
CEO & Co-Founder, Memfault

François Baldassari

CEO / Co-Founder, Memfault

- Passion: tooling and automation in software engineering
- Previously a Firmware Engineer @ Pebble, Oculus, Sun Microsystems
- Can find my thoughts and content on Memfault's Interrupt blog (interrupt.memfault.com)



pebble®



Hardware - it works



It wasn't always the case!



Jaguar — It's like an expensive, unreliable Dodge.

How did we get there?

Quality engineering:

- Stringent materials requirements and inspections
- Characterization
- Tools
- Processes
- Management focus
- Quality assurance checkpoints
- Root-cause analysis and fast iteration



IoT = New Problems

Families are LOCKED OUT of or INSIDE their homes as [REDACTED] 'smart' security app crashes leaving dozens stranded

- Households up and down the UK were unable to lock or unlock their doors
- [REDACTED] customers were also unable to turn their alarms on or off with sirens wailing
- [REDACTED] Smart Home Living App was down for more than 24 hours due to glitch
- ****Have you been affected by the [REDACTED] crash? Email lara.keay@mailonline.co.uk****

Software is a major driver of defects in IoT devices

- No “input qualification”: lots of vendor code is unaudited and provided “as-is”
- High complexity: connectivity, OS, and other layers are 10^6 lines of code or more!
- Performance less well characterized than e.g. that of a given aluminum alloy

👉 At Pebble, we found that software issues were driving the majority of RMAs



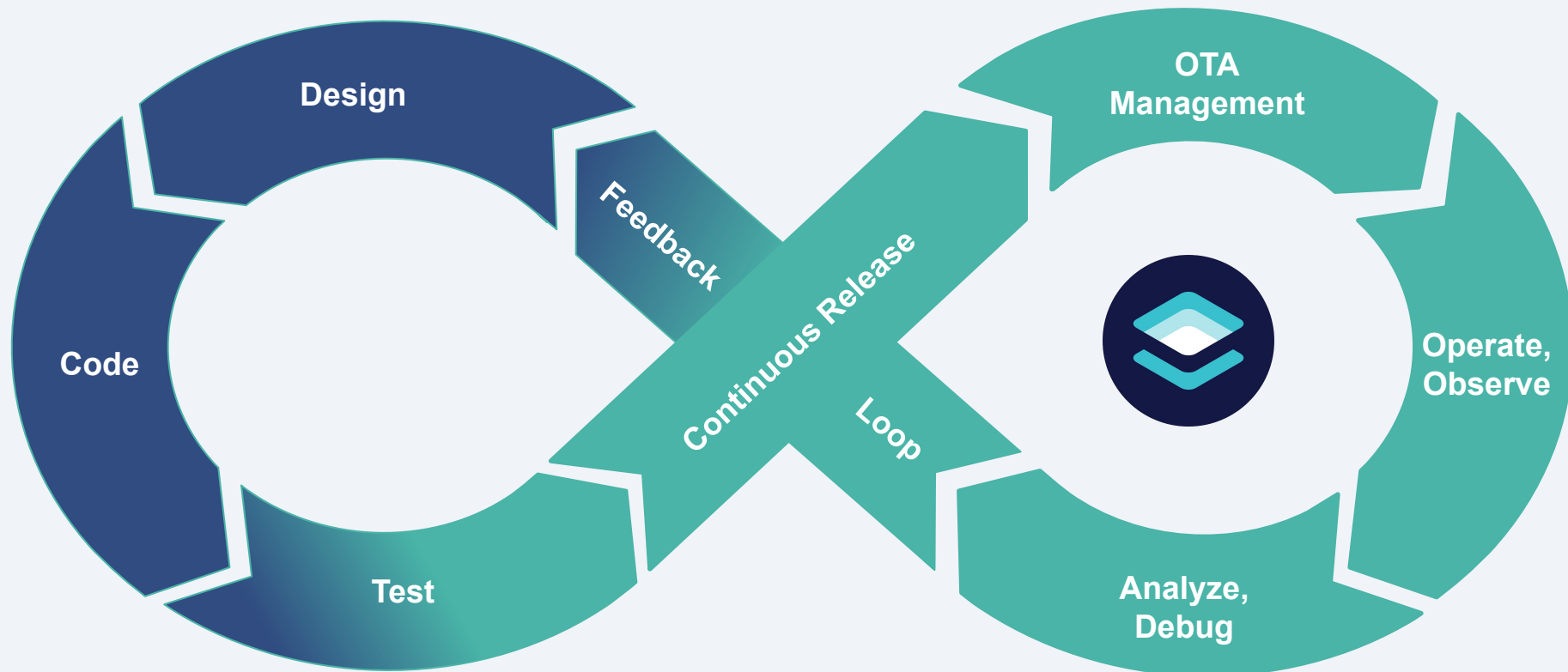
It does not have to be this way!

Software does not rust, decay, wear out, or yield.
It should be our most reliable component!



**Make software the most reliable
part of the IoT.**

Device Reliability Engineering



Improving quality with DRE

**Robust OTA &
DeviceOps**



**Performance
Metrics**



**Remote
Debugging**





Poll #1



Over-the-Air Updates

Software can be improved in the field

Unlike hardware components, software can be improved in the field



This greatly accelerates your rate of iteration



No need to wait for the next manufacturing run!

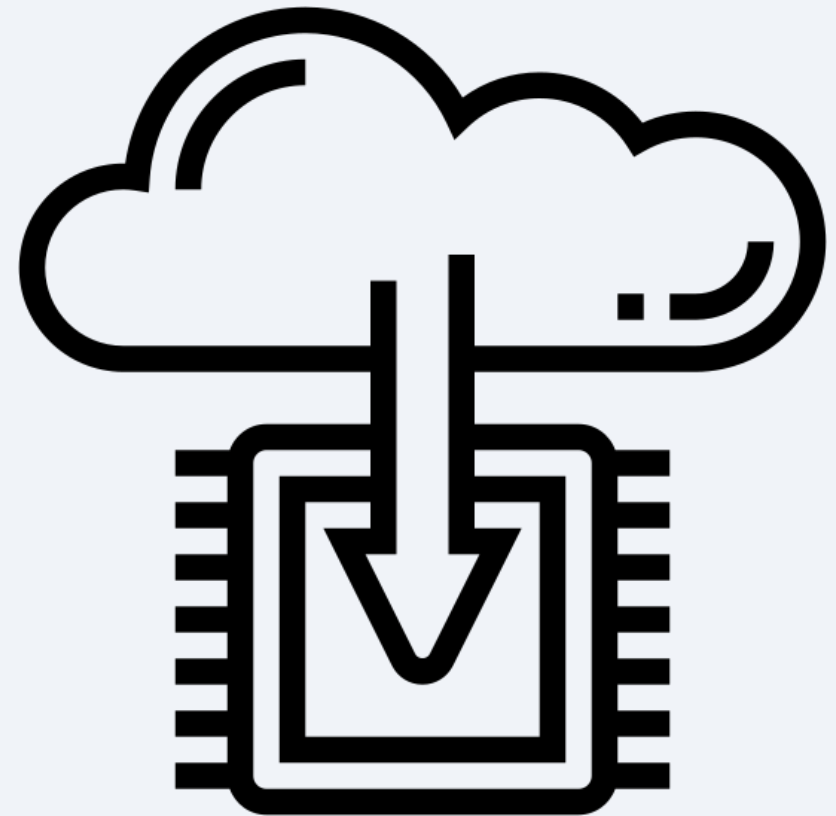


Robust OTA

OTA is your insurance policy against issues

It needs excellent test coverage!

At the very least, your system should support **cohorts, staged rollouts,** and **must-pass-through releases**



Cohorts

What it is

Grouping your devices, and updating each group separately.

Why You Need It

Enable beta tests, A/B tests, and other forms of experimentation for methodical & low-risk release management.

Cohorts with Memfault:

Cohorts		
Cohort	Devices	Release
Beta beta	14	No Release ✎
default	0	No Release ✎
Internal internal	4	0.9.0 ✎
Production prod	18	1.0.0 ✎

Staged Rollouts

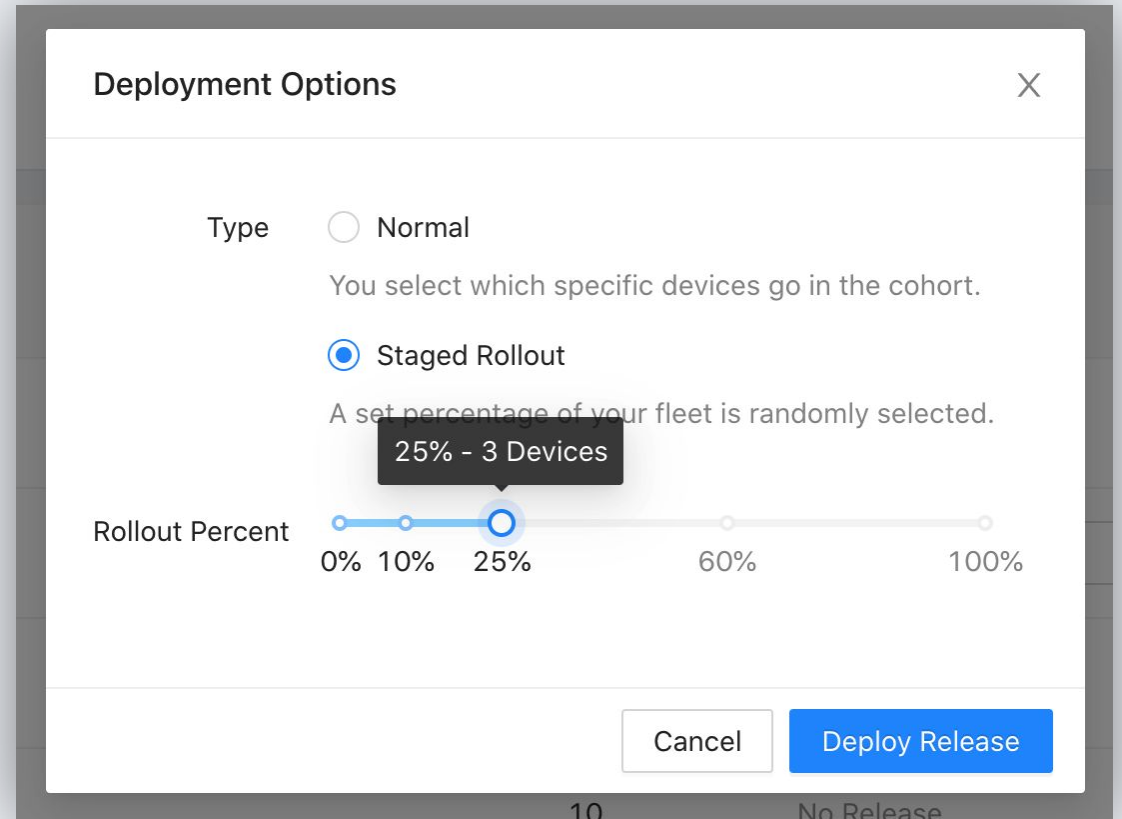
What it is

The ability to roll out a new release to an incrementally larger subset of the fleet.

Why You Need It

Every release introduces risk. By rolling out updates incrementally, you limit the blast radius of any new issue that comes up.

Staged rollouts with Memfault:



The screenshot shows a 'Deployment Options' dialog box with a close button (X) in the top right corner. Under the 'Type' section, there are two radio button options: 'Normal' (unselected) and 'Staged Rollout' (selected). Below 'Normal' is the text 'You select which specific devices go in the cohort.' Below 'Staged Rollout' is the text 'A set percentage of your fleet is randomly selected.' Below this text is a slider control for 'Rollout Percent' ranging from 0% to 100%. The slider has major ticks at 0%, 10%, 25%, 60%, and 100%. A blue circle on the slider is positioned at the 25% mark, and a tooltip above it displays '25% - 3 Devices'. At the bottom of the dialog, there are two buttons: 'Cancel' and 'Deploy Release'.

Must-Pass-Through

What it is

A release which must be loaded on the device before future releases can be installed.

Why You Need It

Gives you control over the upgrade path for complex migrations that may not be forward-compatible.

*Ex. upgrading from 1.2 to 3.8 might require multiple steps:
1.2 → 2.0 → 3.0 → 3.8*

Must-pass-through with Memfault:

Create Full Release

* Version

1.5.0

Revision

Revision for release in backing Version Control System

Notes

This release implements a migration from 1.x firmware to 2.x firmware. All devices must first upgrade to 1.5.0 before they can upgrade to 2.0.0

Must Pass Through

When checked and the [release is activated](#), a device on a lower version will be forced to "pass through" this release before an update is allowed to a release with a version greater than this one. Typically this setting is not needed.

Cancel Create



Performance Metrics

Performance Metrics

“How are my devices doing?”

- ◇ Connectivity
- ◇ Battery Life
- ◇ Memory Usage
- ◇ Sensor Performance
- ◇ System Responsiveness

This system must be:

1. Low overhead (no device impact)
2. Easy to extend
3. Privacy preserving

Individual Device Metrics

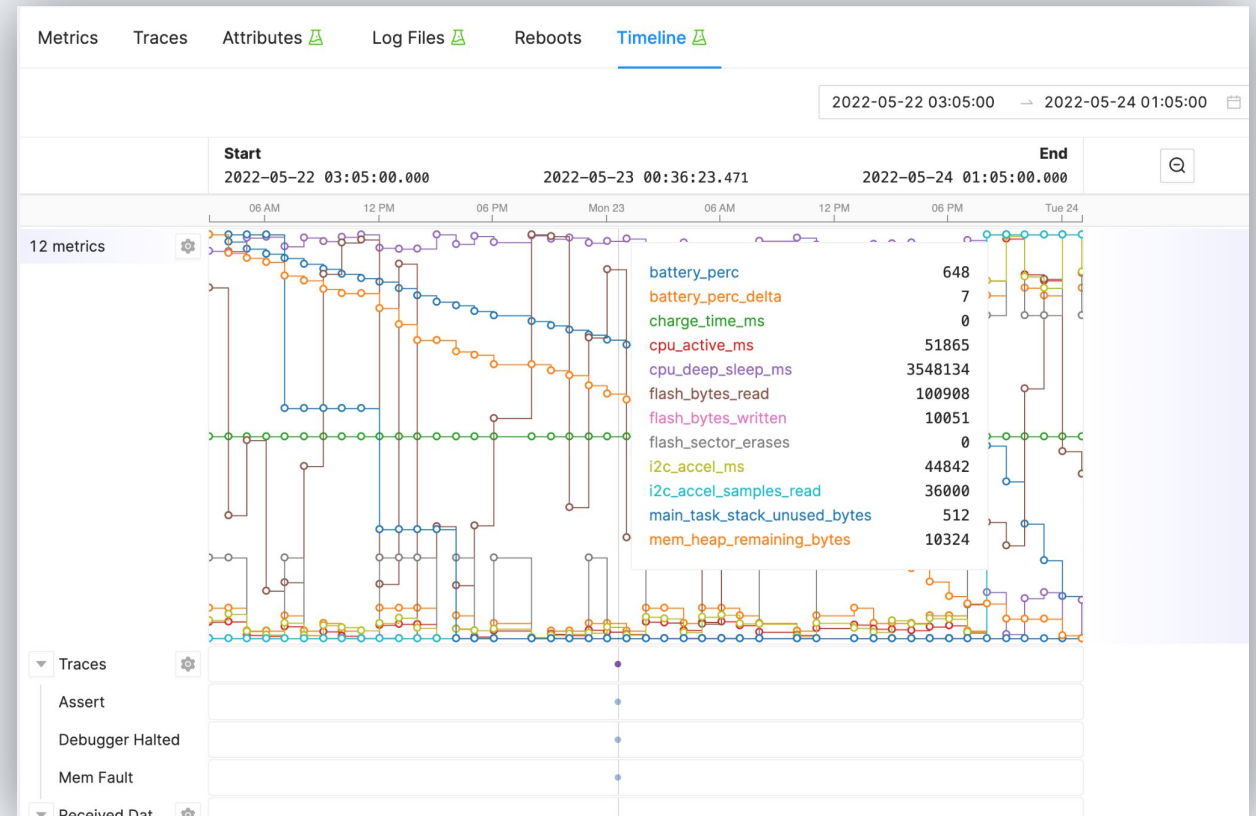
What it is

Collection of data points from devices at regular intervals.

Why You Need It

For customer support or engineering teams to investigate specific reports of devices misbehaving.

Device Metrics with Memfault:



Aggregates and Dashboards

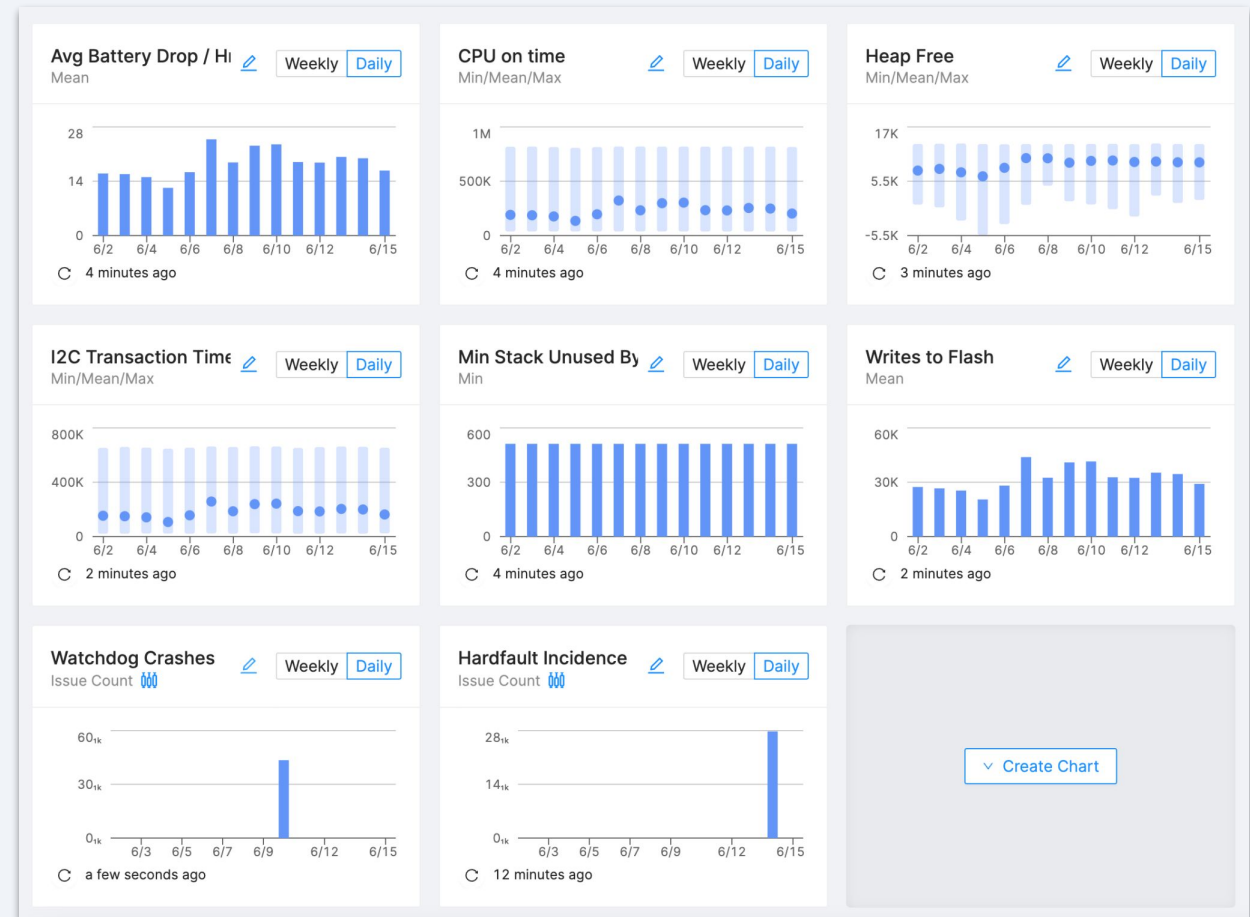
What it is

Dashboards aggregating individual data into high-level charts.

Why You Need It

To understand overall fleet performance and quickly identify trends in the data.

Dashboards with Memfault:



Alerts

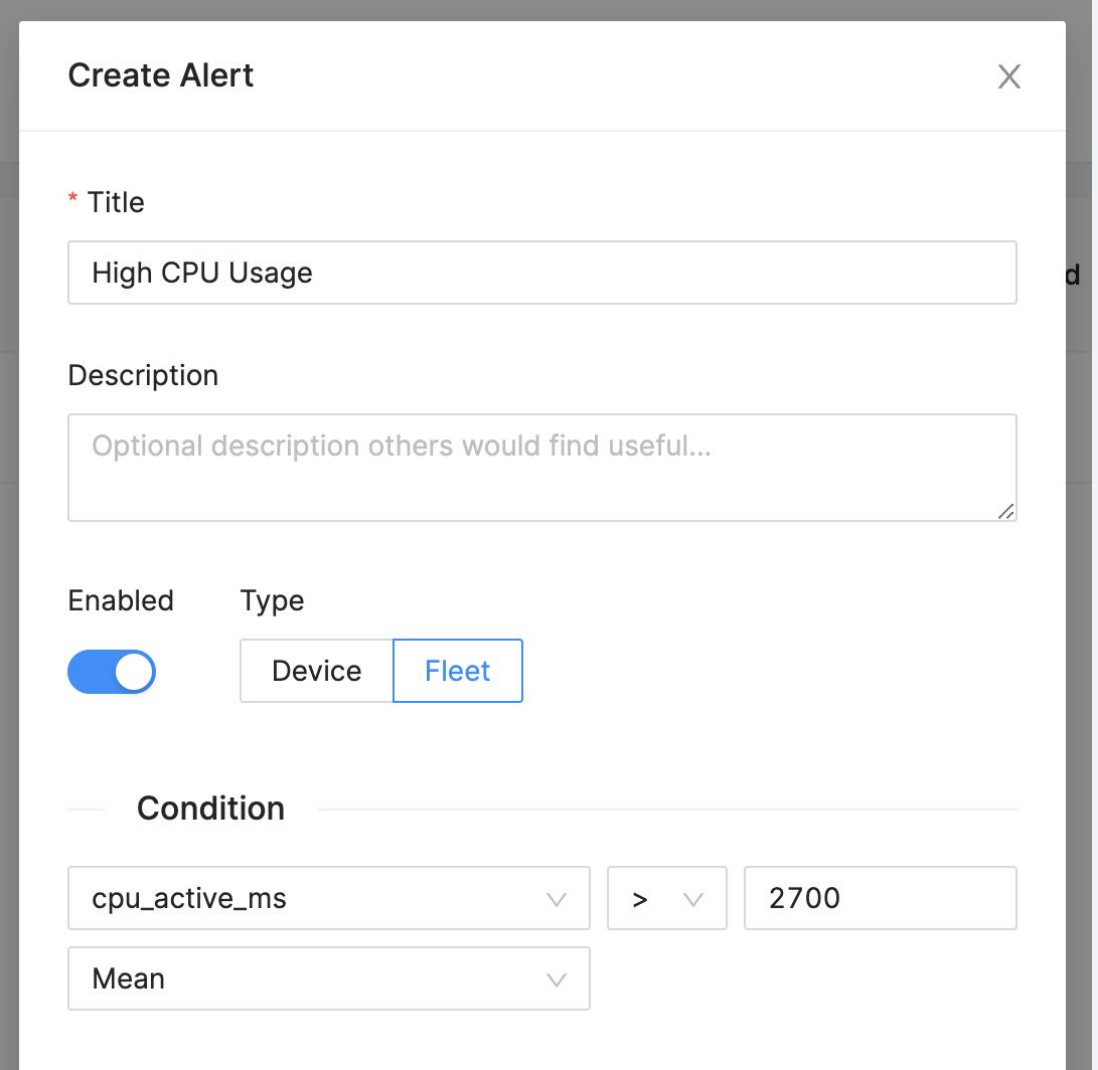
What it is

Alerts to email, slack or incident management platforms when certain conditions are met

Why You Need It

To bring issues to your team's attention quickly, so you can act fast rather than wait for a team member to view the charts.

Alerts with Memfault:



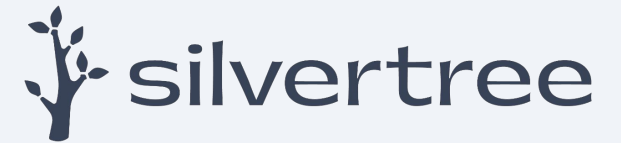
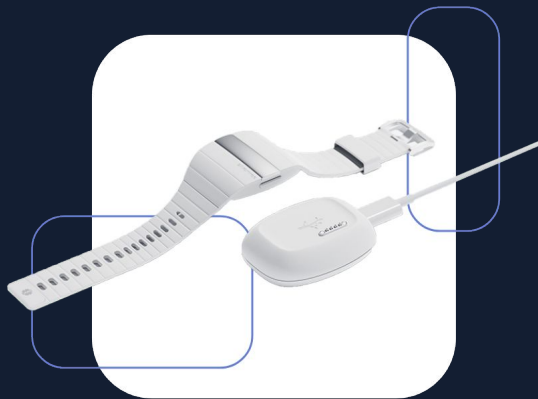
The screenshot shows a 'Create Alert' dialog box with the following fields and options:

- Title:** A text input field containing 'High CPU Usage'.
- Description:** A text area with a placeholder 'Optional description others would find useful...'.
- Enabled:** A toggle switch that is currently turned on.
- Type:** A button group with 'Device' and 'Fleet' options; 'Fleet' is selected.
- Condition:** A section with three input fields: a dropdown menu showing 'cpu_active_ms', a comparison operator dropdown showing '>', and a text input field containing '2700'.
- Aggregation:** A dropdown menu showing 'Mean'.

Case Study: Silvertree

◆ Challenges

- Write-worn, battery powered safety device for older adults
- Complex trade-offs in design between LTE, GPS, and CPU on time vs. battery life



◆ Results

- Monitoring battery level, memory usage, and other metrics enabled Silvertree to make product decisions on key features, like **GPS**
- Data collected by Memfault revealed that Silvertree could reduce GPS on-time by **50%**, saving battery but without jeopardising performance.



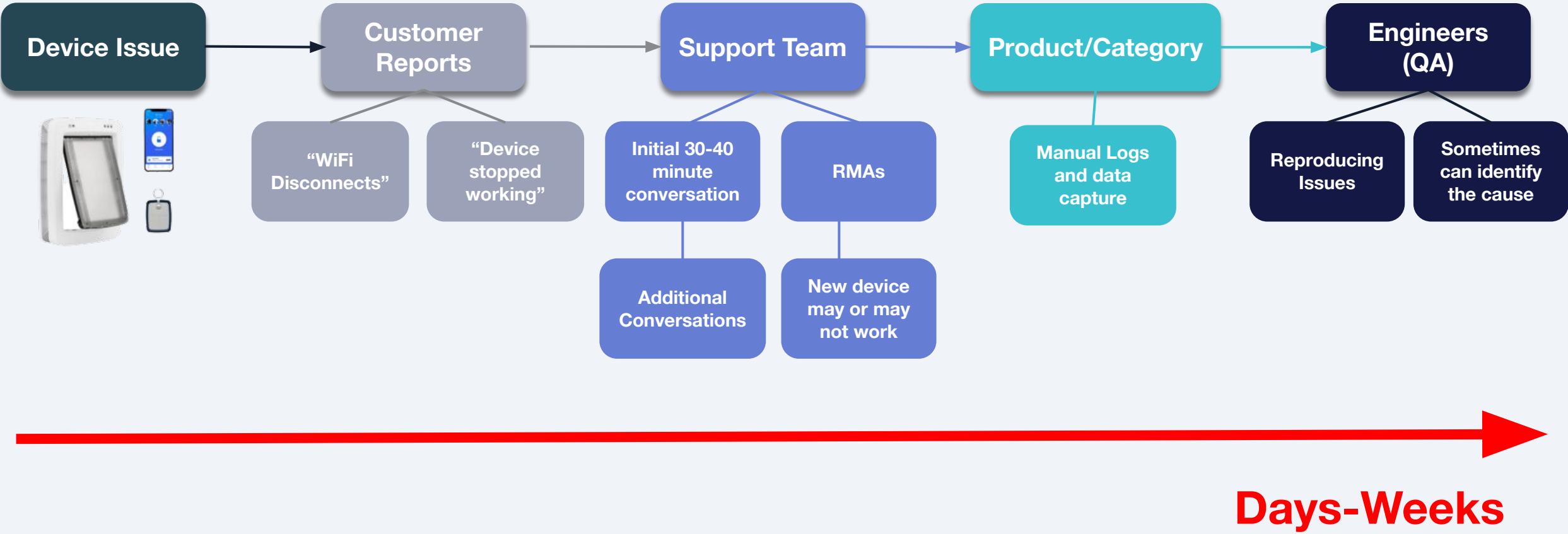
To this date, integrating Memfault was probably one of the best decisions I've made for this company because it saved us so many headaches and so much time.

Konstantin Klitenik
Head of Engineering



Remote Debugging

Traditional methods slow & labor intensive



... and fail to catch 1/10000h bugs

- Bug occurs once every 10,000 hours
- Takes **416 days** to see it on a single device

*With 10,000 devices that issue is hit **every hour***

*With 1 million devices...**every 36 seconds***

Catastrophic issues might *never* be seen during internal testing



Coredumps

What it is

Automatically collect detailed diagnostic data as soon as an issue occurs.

Why You Need It

Give your engineers the information they need to resolve the problem quickly, without an RMA or sending out a technician.

Coredumps with Memfault:

Smart Sink / Issues

Mem Fault at compute_fft [Stack Overflow in accel-workq] [🔗](#) Mem Fault

[Resolve](#) [Merge](#)

[Details](#) [All traces](#) [Comments](#) [Merged issues](#) First Seen Last Seen Devices Traces

a day ago a day ago 125 144

Device [DEMO SERIALNUMBER](#)
Cohort [default](#)
Software [1.0.0-md5+a1c641ba \(main\)](#)
Hardware [DEVBOARD](#)

Older Newer

Captured a day ago

State [Logs](#) [Coredump](#) [Download](#)

[Threads](#) [Exceptions](#) [Registers & Locals](#) [Globals & Sta](#) [Memory Viewer](#)

accel-workq (2) STACK OVERFLOW RUNNING

- 0 compute_fft in .../src/fft.c at line 10
- 1 sleep_algo_compute_sleep_time in .../src/sleep_algo.c at line 12
- 2 process_accel_data_worker_task in .../src/accel_data.c at line 106
- 3 z_work_q_main in .../zephyr/lib/os/work_q.c at line 32
- 4 z_thread_entry in .../lib/os/thread_entry.c at line 29
- 5 0xaaaaaaaa

Thread 3 SUSPENDED

idle (4) READY

logging (5) SUSPENDED

net_mgmt (6) BLOCKED

rx_workq (7) BLOCKED

A dft_out = 0x2000a900 <my_stack_area+1344>
L i = 400
A num_samples = 536912536
A raw_samples = 0x3128115f
L tmp = {1, 222, 7, 84}
R \$r0 = long 536912536 (0x2000a298)
R \$r1 = long 1372324912 (0x51cc0430)
R \$r2 = long 1372324919 (0x51cc0437)
R \$r3 = long 536912832 (0x2000a3c0)
R \$r4 = long 536912508 (0x2000a27c)
R \$r5 = long 536914136 (0x2000a8d8)
R \$r6 = long 0 (0x00000000)
R \$r7 = long 536912488 (0x2000a268)
R \$r8 = long 0 (0x00000000)

Find Address

Regions

0x00000000 00e10020 ...
0x00000004 959e0008 ...
0x00000008 259e0008 %...
0x0000000c 819d0008 ...
0x00000010 819d0008 ...
0x00000014 819d0008 ...
0x00000018 819d0008 ...
0x0000001c 819d0008 ...
0x00000020 819d0008 ...
0x00000024 819d0008 ...
0x00000028 819d0008 ...

Case Study: Diamond Kinetics



◆ Challenges

- No crash logs or stack traces on older smart ball and bat products
- Support tickets had minimal detailed information, making it difficult to reproduce the issue locally
- All devices had to be updated at once to fix an issue on a device



◆ Results

- Reduction in resets per device by **90%** in six months
- **Fixed #1 crash**, a connectivity bug in Bluetooth LE chip
- Faster time to market by accelerating the development cycle



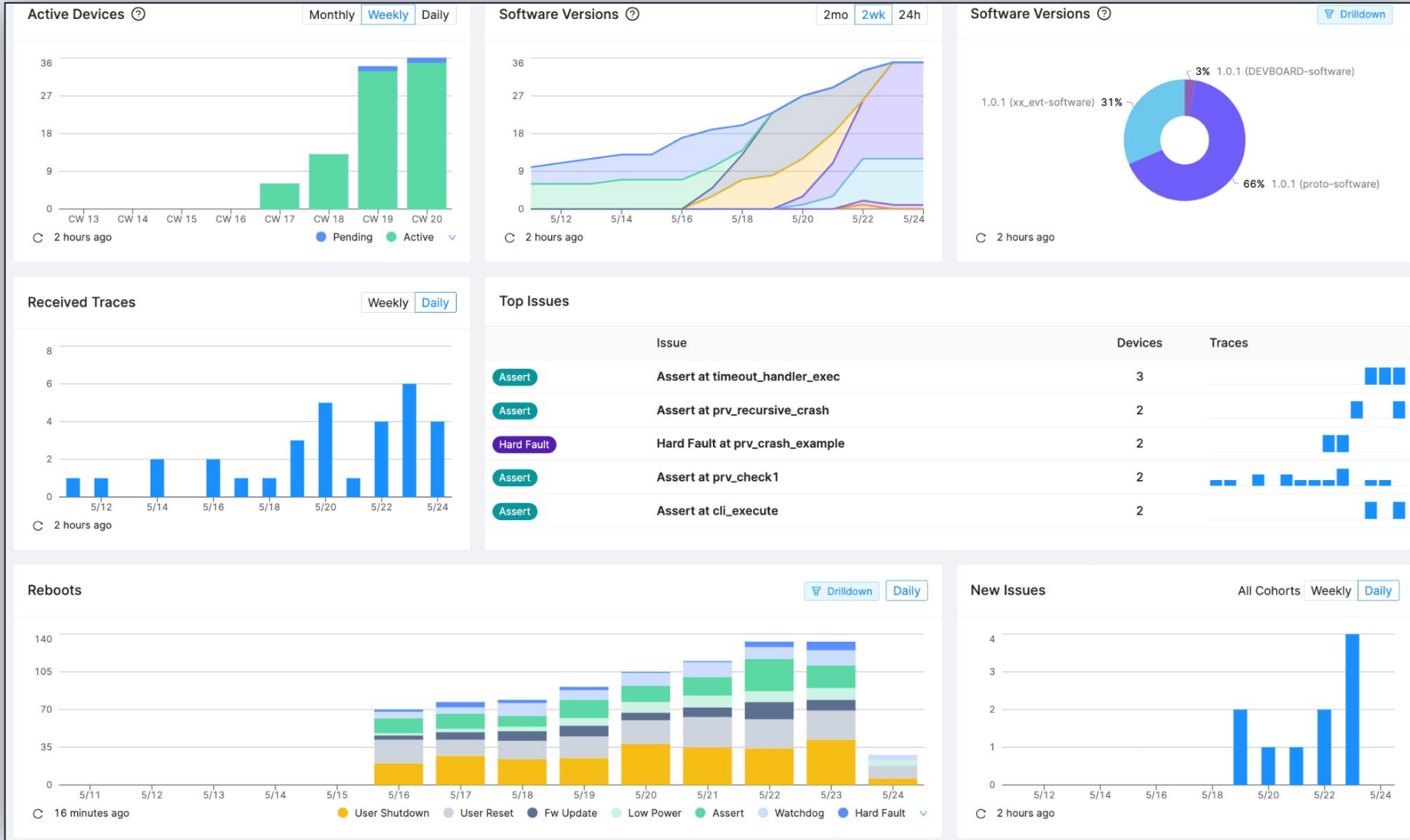
Every step of the way, the Memfault SDK has been reliable and easy to integrate. I wouldn't go to market on an IoT device without Memfault in place.

Mike Ressler
CTO



Poll #2

Memfault: IoT Reliability Platform





Memfault