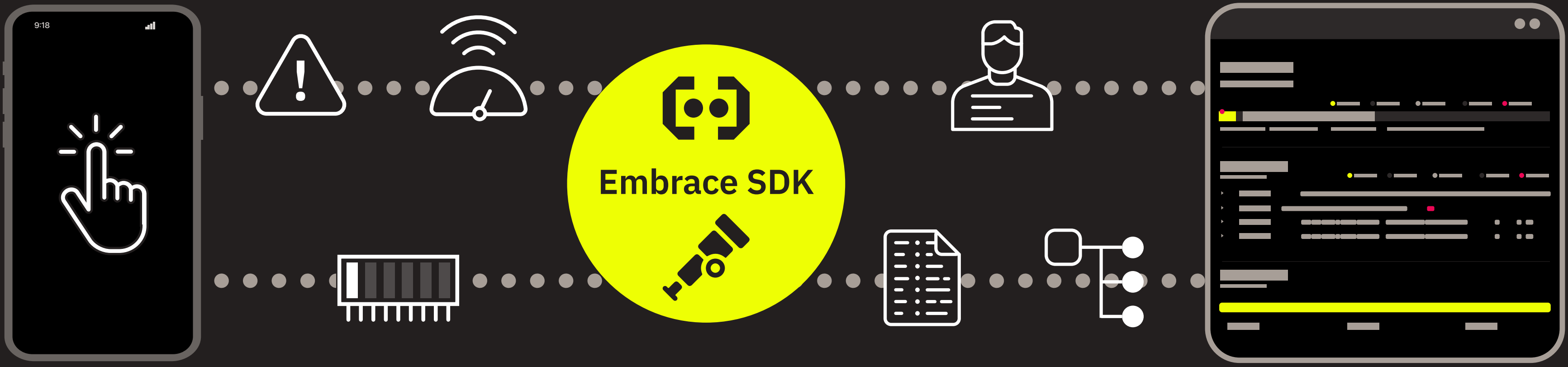
Embrace solution guide



embrace

Modern mobile observability built on OpenTelemetry

Embrace combines open-source SDKs with an analysis dashboard to help the entire engineering team understand exactly what is disrupting mobile user experiences.

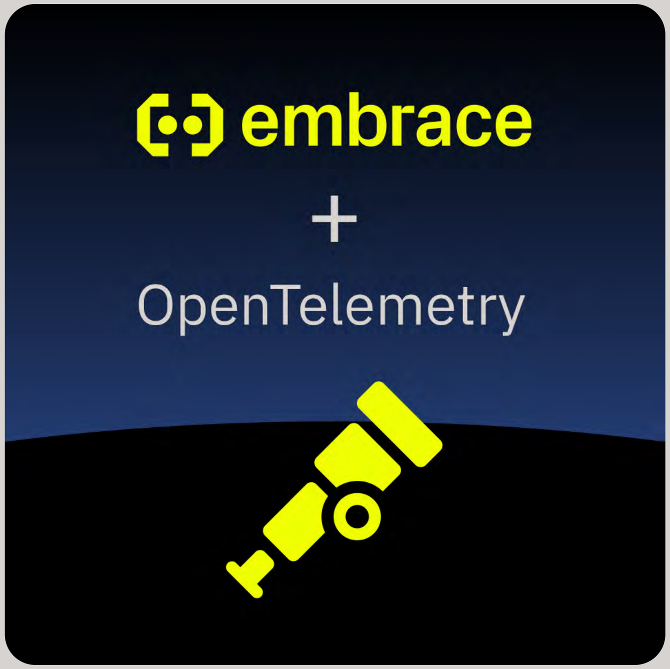


Table of contents

embrace

The challenge with current mobile observability practices

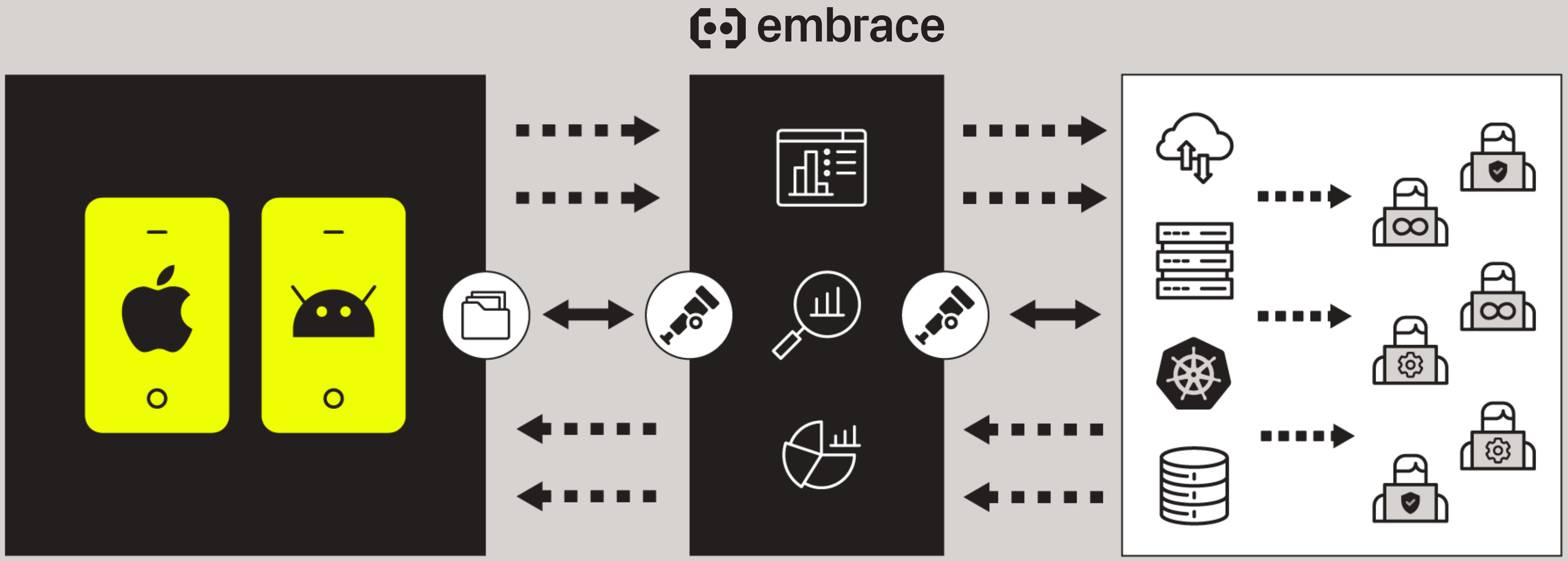
While the importance of mobile apps only continues to grow, the maturity of mobile observability practices in organizations has not kept pace. There are a few key reasons for this:

- Mobile teams frequently don’t build observability into their development practices, having mostly worked with crash reporting and error monitoring tools.
- Mobile observability solutions, by and large, do not interoperate well with existing backend observability solutions.
- There hasn’t been a shared observability language to allow collaboration across frontend and backend teams.

This is a fundamental problem for modern organizations that want their observability practices to connect all the way from the data center to the user interface so they can understand the quality and business impact of their end-user experiences.

So, how can organizations effectively modernize their mobile observability practices?

To answer that, Embrace has created the only user-focused, mobile-first observability solution built on OpenTelemetry. By delivering crucial performance insights across DevOps and mobile teams, Embrace illuminates real customer impact—not just server impact—to deliver the best user experiences.

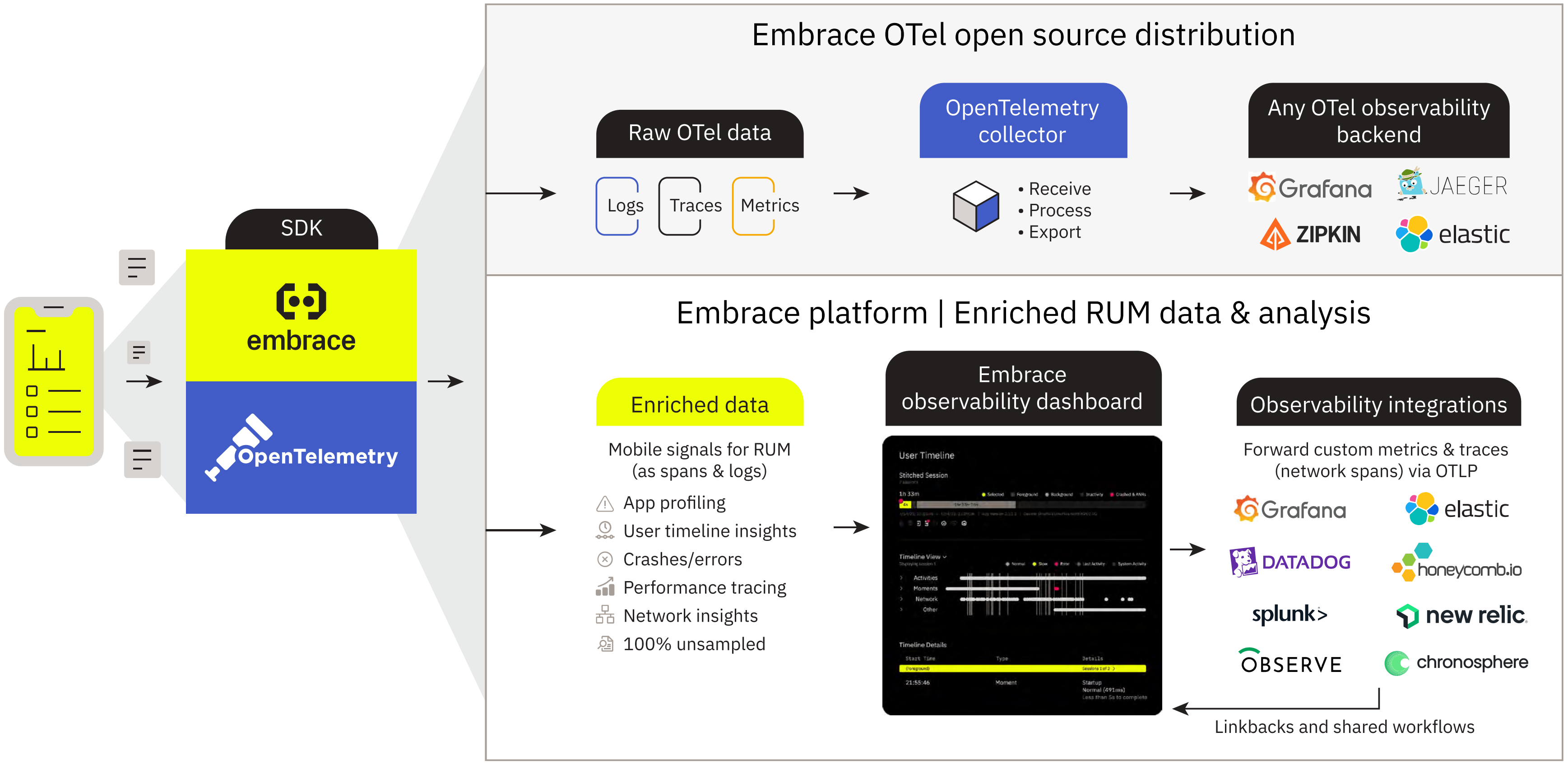


What is Embrace?

Embrace is an observability tool, combining open-source SDKs with an analysis dashboard that, together, enable the entire engineering team to understand the full picture of what is disrupting mobile user experiences, such as:

- Are users abandoning apps during startup because of slow network calls?
- Are third-party SDKs causing issues like crashes and Application Not Responding (ANR) errors?
- Are key user actions like login, add to cart, or checkout running slower than expected or outright failing?
- Are users on certain devices experiencing memory issues from loading too many images and videos?
- Are users experiencing issues when trying to complete tasks across multiple sessions?

Mobile teams integrate the Embrace OpenTelemetry-compliant SDK – supported platforms include iOS, Android, React Native, Unity, and Flutter – and then can access high-fidelity data from every user session in the Embrace analytics and observability dashboard. Engineering teams can then use this data for issue discovery and resolution, optimizing app performance, tracking internal metrics, and powering SLOs by integrating with existing observability solutions.



With Embrace's open, composable approach to mobile observability, teams can modernize their observability stack while maintaining flexibility and control over their data. They can:

- Instrument mobile apps once and send data anywhere without vendor lock-in.
- Unify mobile and backend observability for full-stack visibility.
- Correlate user experience with system health for a complete business impact analysis.
- Build independently with open source, OTel-based SDKs, or tap into an enterprise offering with advanced analytics, full technical and behavioral user timelines, and enriched data forwarding.

Let's focus on three key components of Embrace that highlight the value of this modern approach to mobile observability:

- **User-focused observability**
- **Built on OpenTelemetry**
- **Connecting mobile teams and DevOps/SREs**

User-focused observability

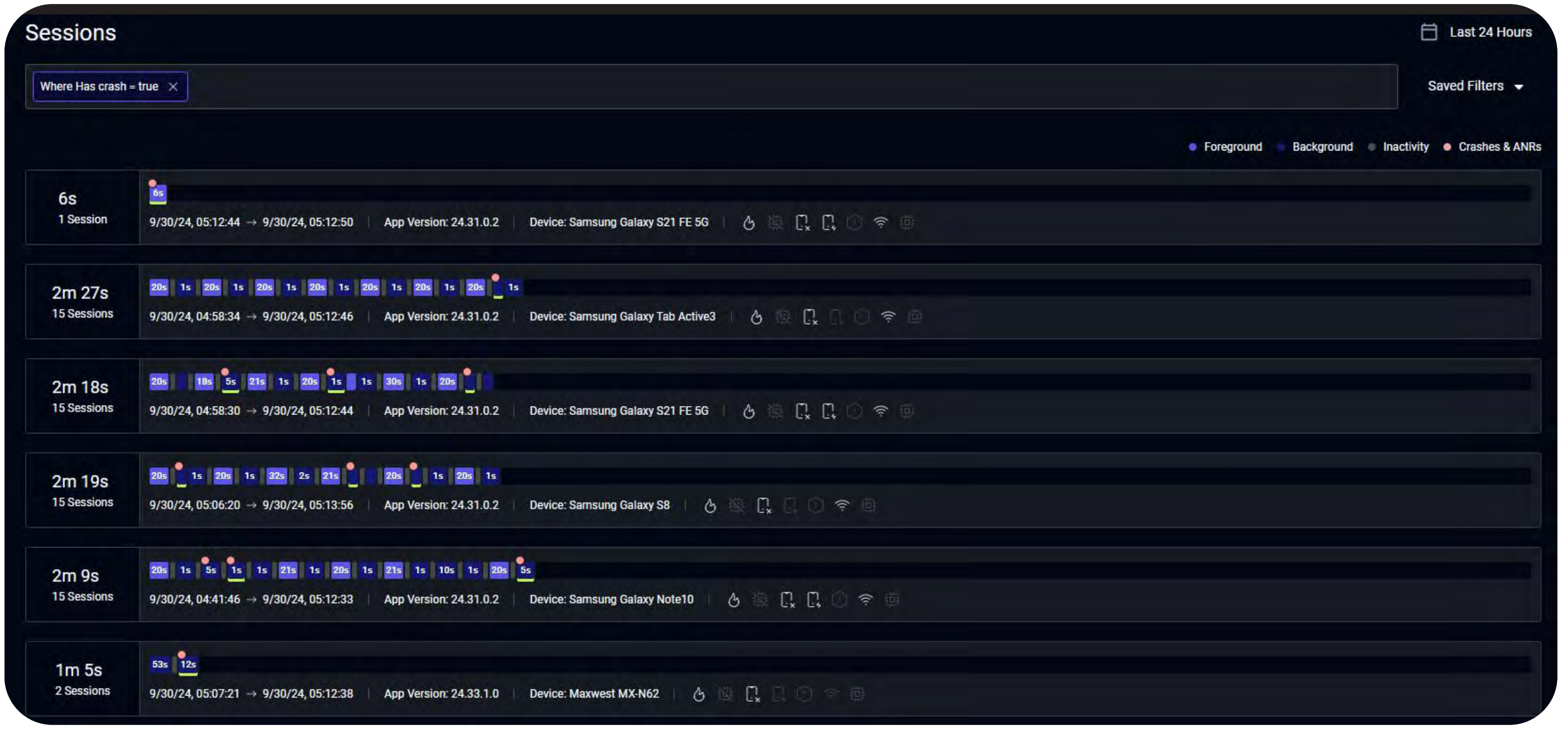
In backend systems, you frequently focus on measuring the latency and failure rate of services. However, this approach doesn't show the full picture of impact on end users. For example:

- Your backend service has a slowdown, but user behavior does not change.
- **Your backend service has a spike in failures, but it's only affecting a small group of users on devices that you're no longer supporting.**
- Your backend service has a spike in failures, but users are able to immediately retry and successfully complete purchases.

The best way to prioritize engineering resources is by understanding the user and business impact of issues. And the only way to do that is by measuring from the perspective of your users.

Every session for every user

Embrace collects every session for every user, including when the app is backgrounded, and visualizes them as timelines, providing a complete picture of the end-user experience. If you receive a user complaint, you can look up the user and review everything that happened while they were using your app in order to track down what went wrong.



6s
1 Session
6s
9/30/24, 05:12:44 → 9/30/24, 05:12:50 | App Version: 24.31.0.2 | Device: Samsung Galaxy S21 FE 5G

2m 27s
15 Sessions
20s 1s 20s 1s 20s 1s 20s 1s 20s 1s 20s 1s 20s 1s
9/30/24, 04:58:34 → 9/30/24, 05:12:46 | App Version: 24.31.0.2 | Device: Samsung Galaxy Tab Active3

2m 18s
15 Sessions
20s 18s 5s 21s 1s 20s 1s 1s 30s 1s 20s
9/30/24, 04:58:30 → 9/30/24, 05:12:44 | App Version: 24.31.0.2 | Device: Samsung Galaxy S21 FE 5G

2m 19s
15 Sessions
20s 1s 20s 1s 32s 2s 21s 20s 1s 20s 1s
9/30/24, 05:06:20 → 9/30/24, 05:13:56 | App Version: 24.31.0.2 | Device: Samsung Galaxy S8

2m 9s
15 Sessions
20s 1s 5s 1s 1s 21s 1s 20s 1s 21s 1s 10s 1s 20s 5s
9/30/24, 04:41:46 → 9/30/24, 05:12:33 | App Version: 24.31.0.2 | Device: Samsung Galaxy Note10

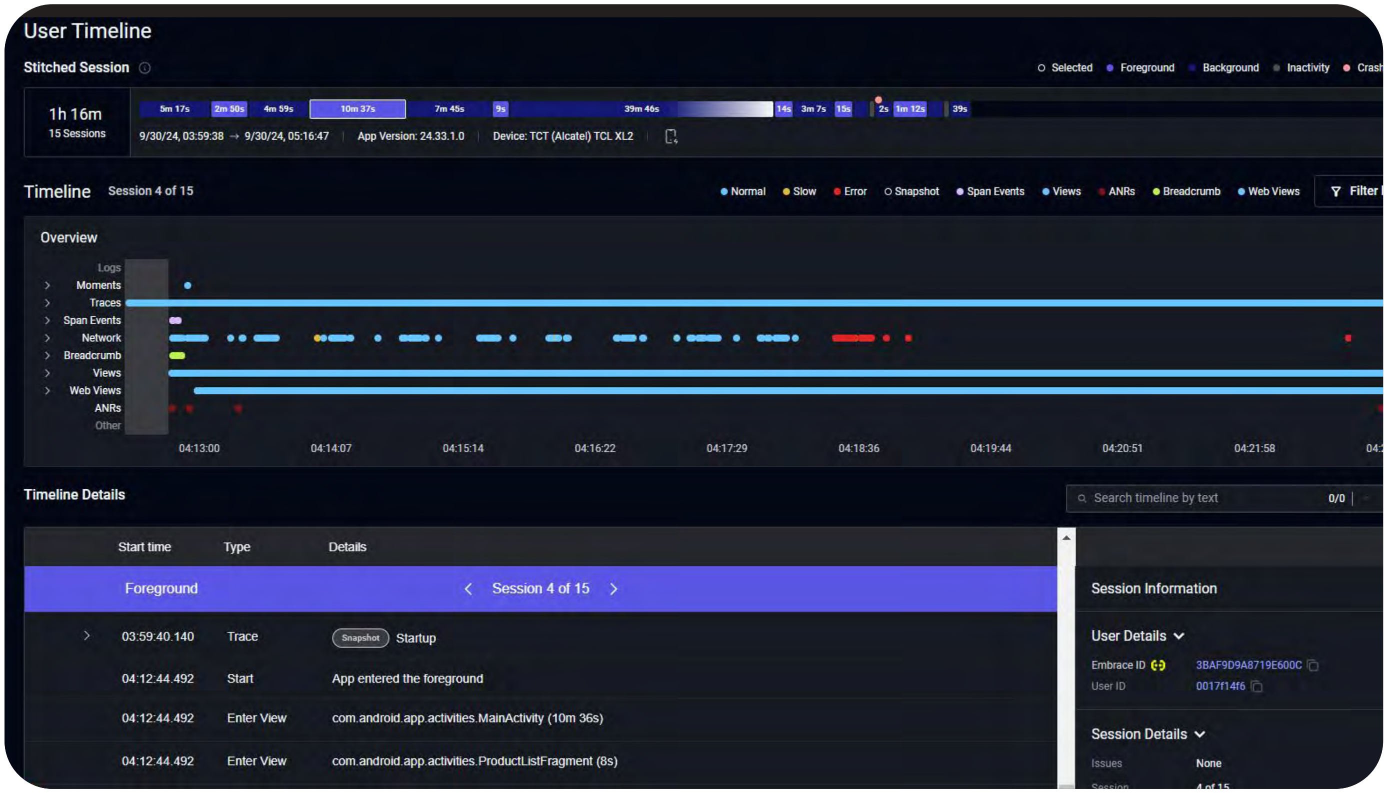
1m 5s
2 Sessions
53s 12s
9/30/24, 05:07:21 → 9/30/24, 05:12:38 | App Version: 24.33.1.0 | Device: Maxwest MX-N62

In the above visual, we're filtering for crashes from the Sessions page, and we can see every session where a user experienced a crash. We can also filter by any combination of issue, device, OS, app state, log, and span variables.

embrace

Full context for every session

When we want to dive into an individual session, we can load the User Timeline, which shows the full technical and behavioral details. We can see where slow and failing network calls, key user flows, breadcrumbs, device state changes, and more happen as a user navigates the app. With this context, mobile engineers can easily troubleshoot key mobile app issues, like crashes, errors, ANRs, out of memory crashes (OOMs), and user terminations.



9/30/24, 03:59:38 → 9/30/24, 05:16:47 | App Version: 24.33.1.0 | Device: TCT (Alcatel) TCL XL2

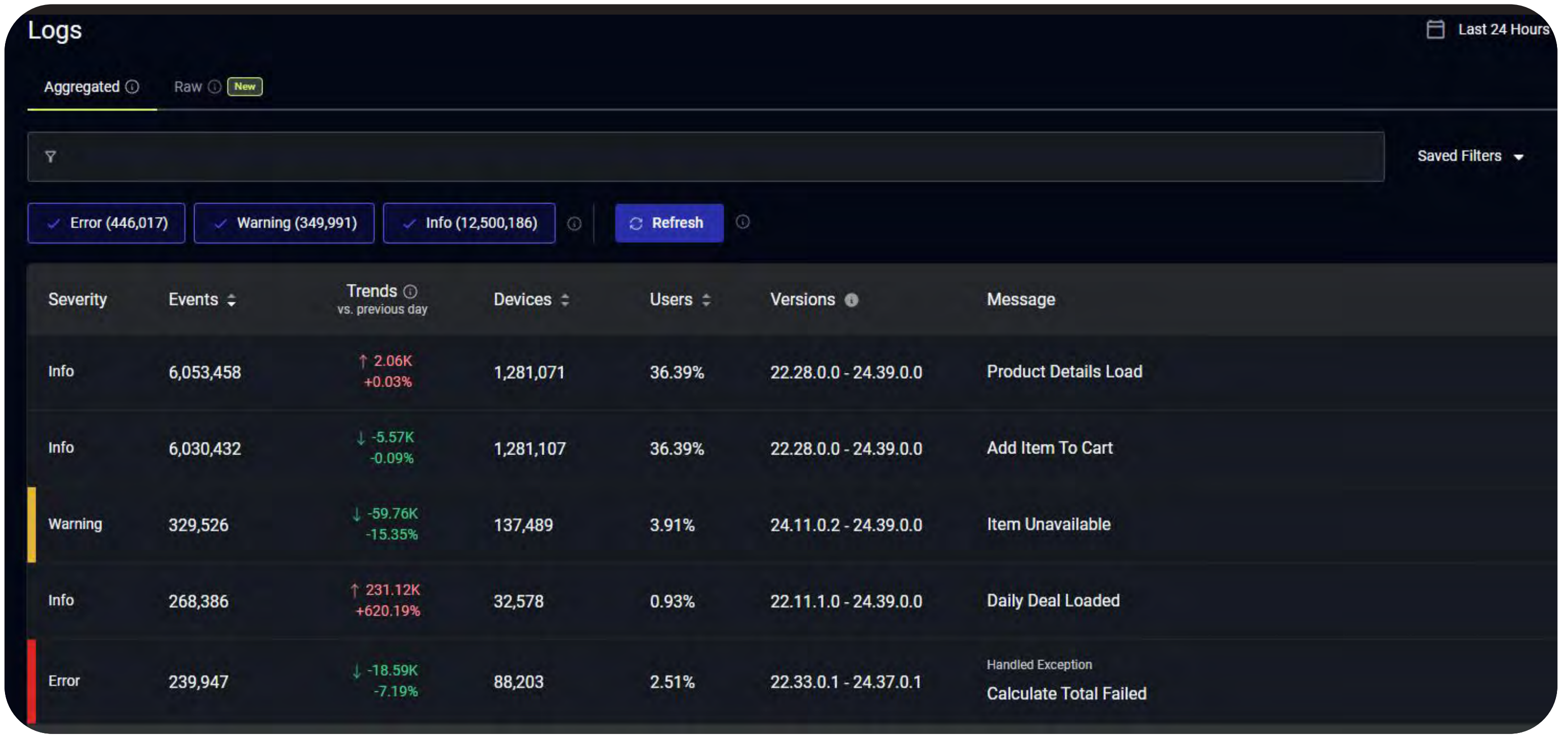
Start time	Type	Details
Foreground		Session 4 of 15
03:59:40.140	Trace	Snapshot Startup
04:12:44.492	Start	App entered the foreground
04:12:44.492	Enter View	com.android.app.activities.MainActivity (10m 36s)
04:12:44.492	Enter View	com.android.app.activities.ProductListFragment (8s)

Embrace ID	3BAF9D9A8719E600C
User ID	0017f14f6
Issues	None
Session	4 of 15

embrace

Logs

Logs are one of the core signals in observability, and Embrace provides powerful ways to leverage them. You can enrich them with as much context as you want, and then easily use free-text search and advanced filtering to quickly identify the events you're interested in. You can also review your raw logs, or use the aggregated view to see big-picture trends before diving into the individual sessions for debugging.

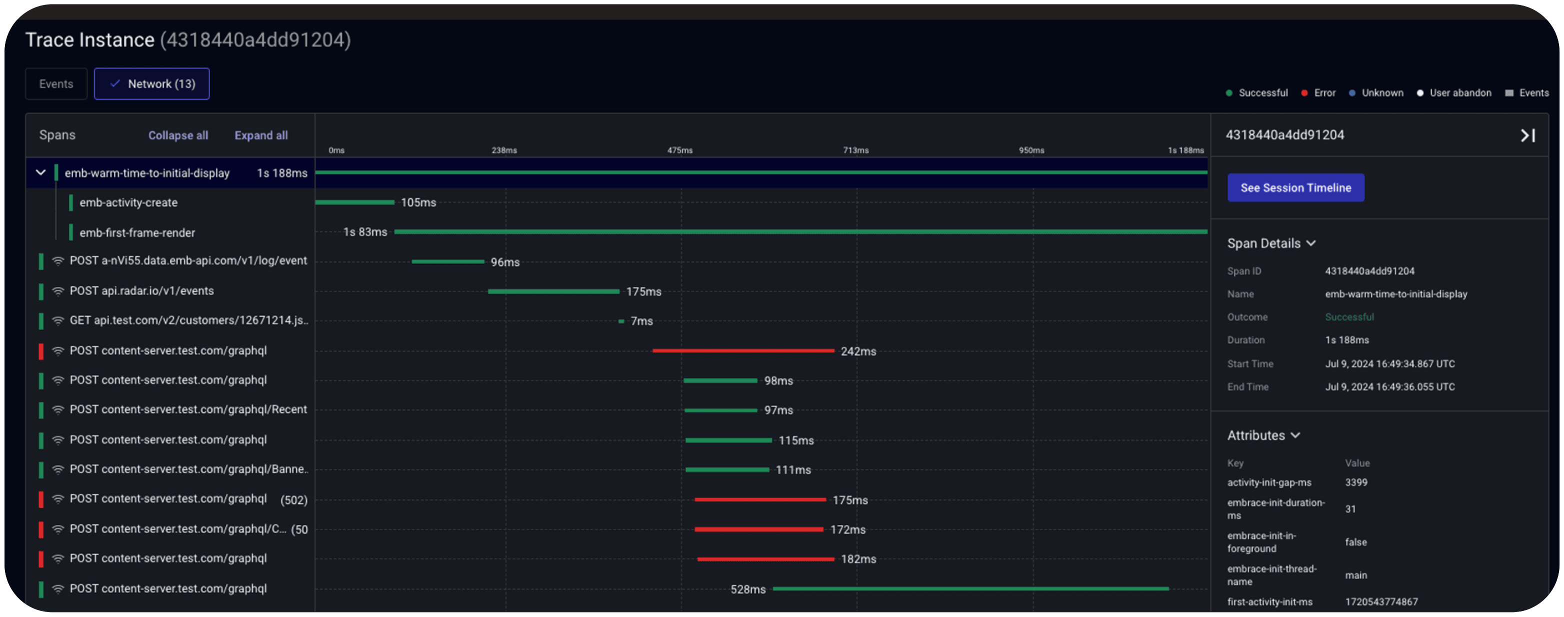


Severity	Events	Trends vs. previous day	Devices	Users	Versions	Message
Info	6,053,458	↑ 2.06K +0.03%	1,281,071	36.39%	22.28.0.0 - 24.39.0.0	Product Details Load
Info	6,030,432	↓ -5.57K -0.09%	1,281,107	36.39%	22.28.0.0 - 24.39.0.0	Add Item To Cart
Warning	329,526	↓ -59.76K -15.35%	137,489	3.91%	24.11.0.2 - 24.39.0.0	Item Unavailable
Info	268,386	↑ 231.12K +620.19%	32,578	0.93%	22.11.1.0 - 24.39.0.0	Daily Deal Loaded
Error	239,947	↓ -18.59K -7.19%	88,203	2.51%	22.33.0.1 - 24.37.0.1	Handled Exception Calculate Total Failed

embrace

Tracing

You can instrument spans and traces to monitor key user actions in your mobile app, like login, search, and checkout. Embrace also automatically includes all concurrent network requests as spans within your performance traces. This enhanced context helps engineers clearly understand the temporal relationship between network activity and other events, leading to quicker insights and faster issue resolution.



Spans	Duration
emb-warm-time-to-initial-display	1s 188ms
emb-activity-create	105ms
emb-first-frame-render	1s 83ms
POST a-nVi55.data.emb-api.com/v1/log/event	96ms
POST api.radar.io/v1/events	175ms
GET api.test.com/v2/customers/12671214.js..	7ms
POST content-server.test.com/graphql	242ms
POST content-server.test.com/graphql	98ms
POST content-server.test.com/graphql/Recent	97ms
POST content-server.test.com/graphql	115ms
POST content-server.test.com/graphql/Banne..	111ms
POST content-server.test.com/graphql (502)	175ms
POST content-server.test.com/graphql/C... (50	172ms
POST content-server.test.com/graphql	182ms
POST content-server.test.com/graphql	528ms

4318440a4dd91204

Span Details

Span ID	4318440a4dd91204
Name	emb-warm-time-to-initial-display
Outcome	Successful
Duration	1s 188ms
Start Time	Jul 9, 2024 16:49:34.867 UTC
End Time	Jul 9, 2024 16:49:36.055 UTC

Attributes

Key	Value
activity-init-gap-ms	3399
embrace-init-duration-ms	31
embrace-init-in-foreground	false
embrace-init-thread-name	main
first-activity-init-ms	1720543774867

Built on OpenTelemetry

We're the first mobile-only vendor in the Cloud Native Computing Foundation (CNCF), which oversees the OpenTelemetry project, and our SDKs are open source and built on OpenTelemetry. That means anyone can integrate the Embrace SDKs and send mobile telemetry to any OTel-compatible backend.

Want to try it for yourself? Check out our iOS and Android OpenTelemetry exporter walkthroughs where you'll learn how to send spans directly from the Embrace SDK to Zipkin.

We've also updated our data models to fit OpenTelemetry wherever possible, and for instances where mobile-specific issues prevent this – such as dealing with in-progress spans that cross lifecycle boundaries – we're collaborating with the community to help move the standards forward as well as contribute our solutions back into the OTel standards.

If you'd like to learn more about the challenges in modeling mobile telemetry in OpenTelemetry, check out our OTel Community Day talk.

Connecting mobile teams and DevOps/SREs

In order to prioritize engineering resources effectively, your org needs to be able to quickly isolate an issue and assign a fix, regardless of where it's originating across your tech stack.

When it comes to mobile apps, however, the problem has historically been that frontend and backend teams have struggled to collaborate, due to data silos and a lack of a shared observability language. This often leads to a much lengthier, more difficult issue resolution process that ends up impacting the end user. Here are a few examples where this disconnect causes significant toil:

- A backend service that powers login has an increase in latency, but the DevOps team doesn't know the user impact. The mobile team can't look up whether users are abandoning during login because their tool only measures crashes and errors.

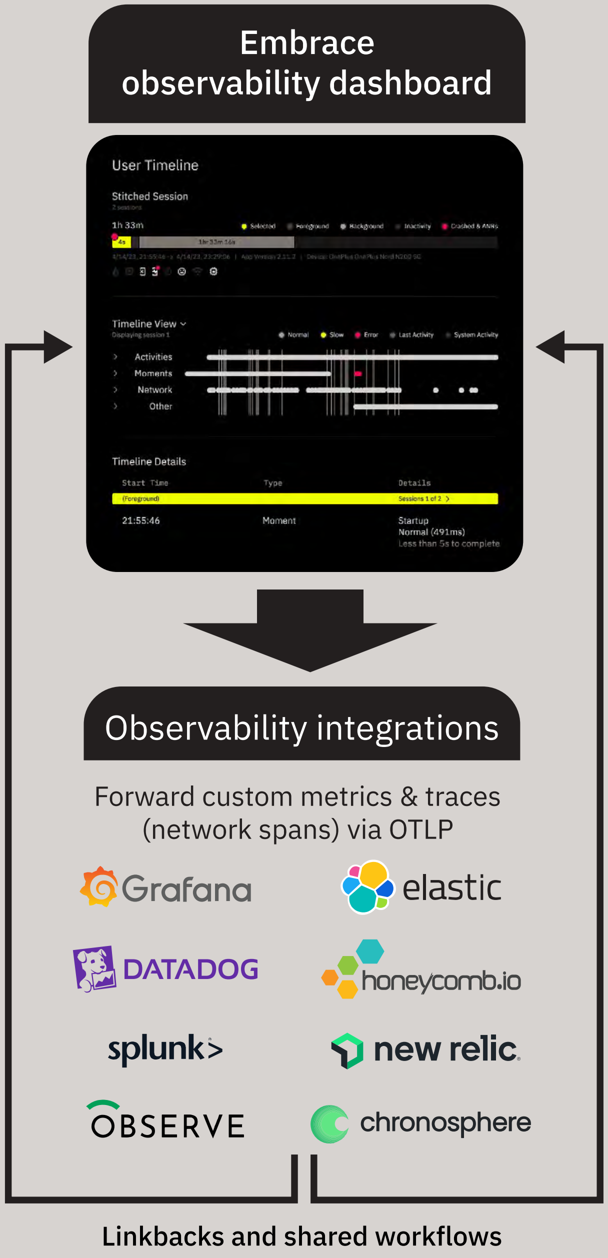
- User complaints increase about search functionality not working properly. The mobile team suspects something's wrong with the search service, but the DevOps team says everything is fine on their end. The mobile team does not have visibility to resolve the issue.

With user-focused observability like Embrace provides, the mobile team could look up the user abandonment for the login functionality, and they could also tell which searches were processed successfully for users. But what about the visibility gap between the frontend and backend?

Wouldn't it be much better if mobile signals could be forwarded to the DevOps team, so that they could always see the user impact data alongside the service health data?

Even more powerful, what if the DevOps team could connect directly from a failure in the backend to the affected user experience in the mobile app?

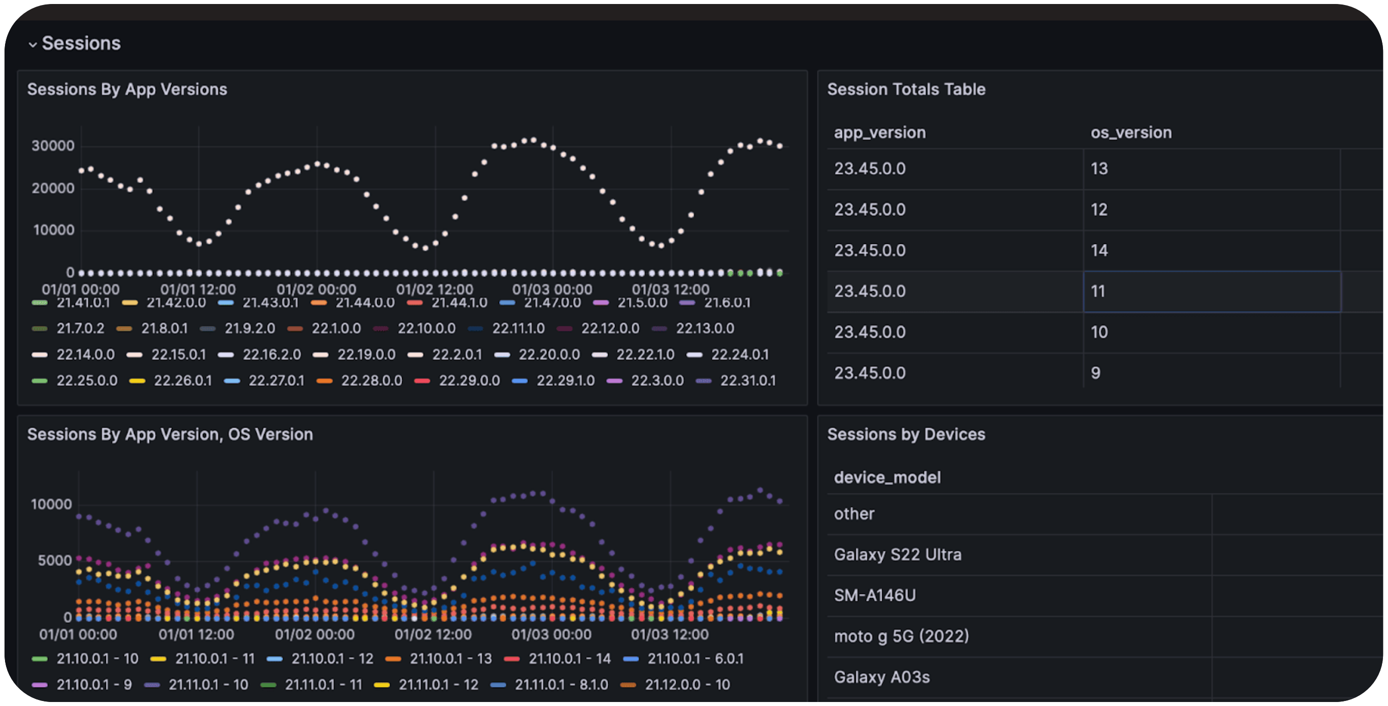
With Embrace, you can do both!



Sending mobile metrics to the backend

Embrace can [forward custom metrics]() to many popular observability tools, including Grafana, Chronosphere, Honeycomb, New Relic, and more. You can create metrics from any number of sources, including crashes, logs, spans, sessions, and networking. You can also filter them by [many different dimensions](), so that you're measuring exactly what you want. Within Embrace, you're [just a few clicks away]() from forwarding any metric.

The following image highlights [sending Embrace custom metrics to Grafana]().



Session Totals Table

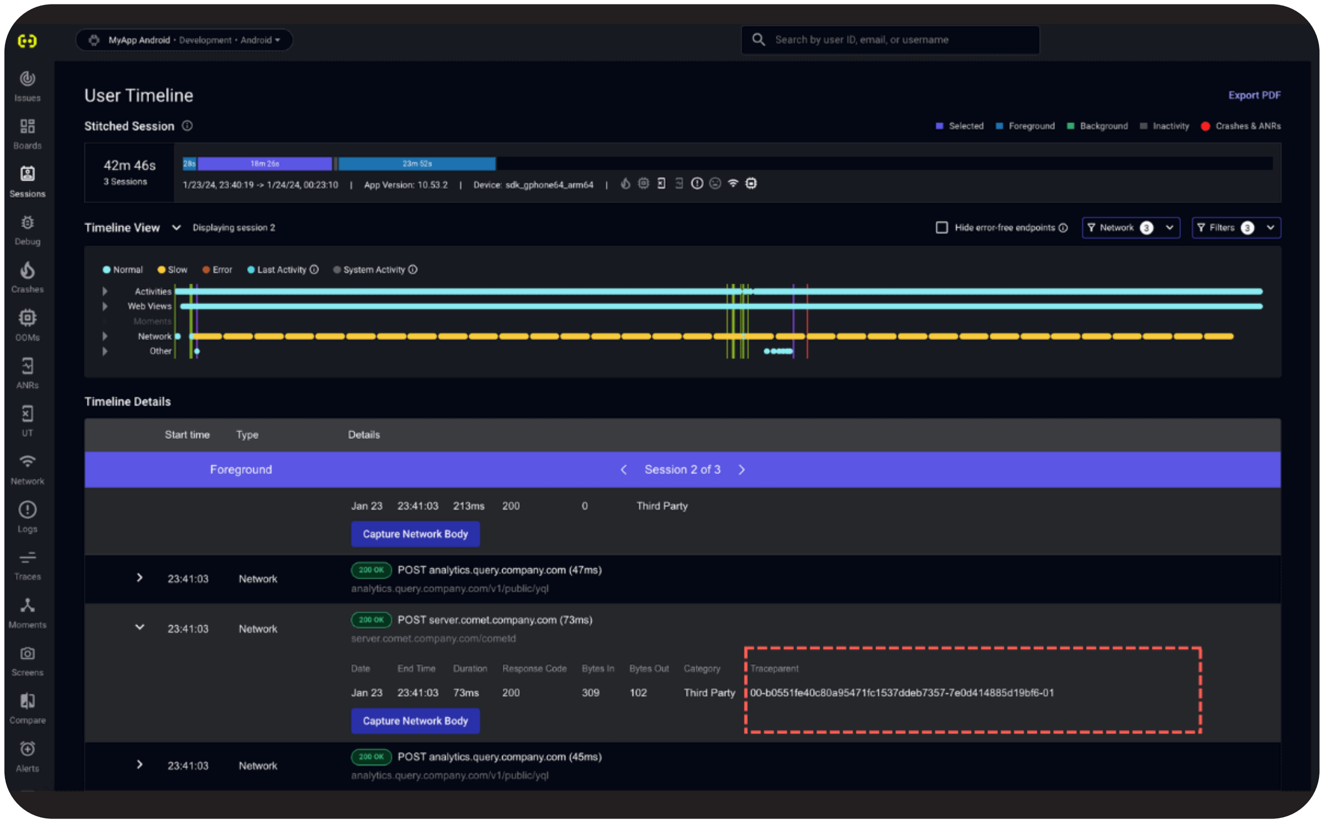
app_version	os_version
23.45.0.0	13
23.45.0.0	12
23.45.0.0	14
23.45.0.0	11
23.45.0.0	10
23.45.0.0	9

Sessions by Devices

device_model
other
Galaxy S22 Ultra
SM-A146U
moto g 5G (2022)
Galaxy A03s

Connecting from the backend to mobile

Embrace has a feature called Network Span Forwarding that works by intercepting a network request on the client side and then injecting a trace parent into it. Your backend monitoring will capture the request with its unique trace ID, and the same trace ID will be tagged to the network request in the corresponding user session in Embrace. That way, when investigating an issue, both the mobile team and the backend team can look up the same network request to understand what went wrong.

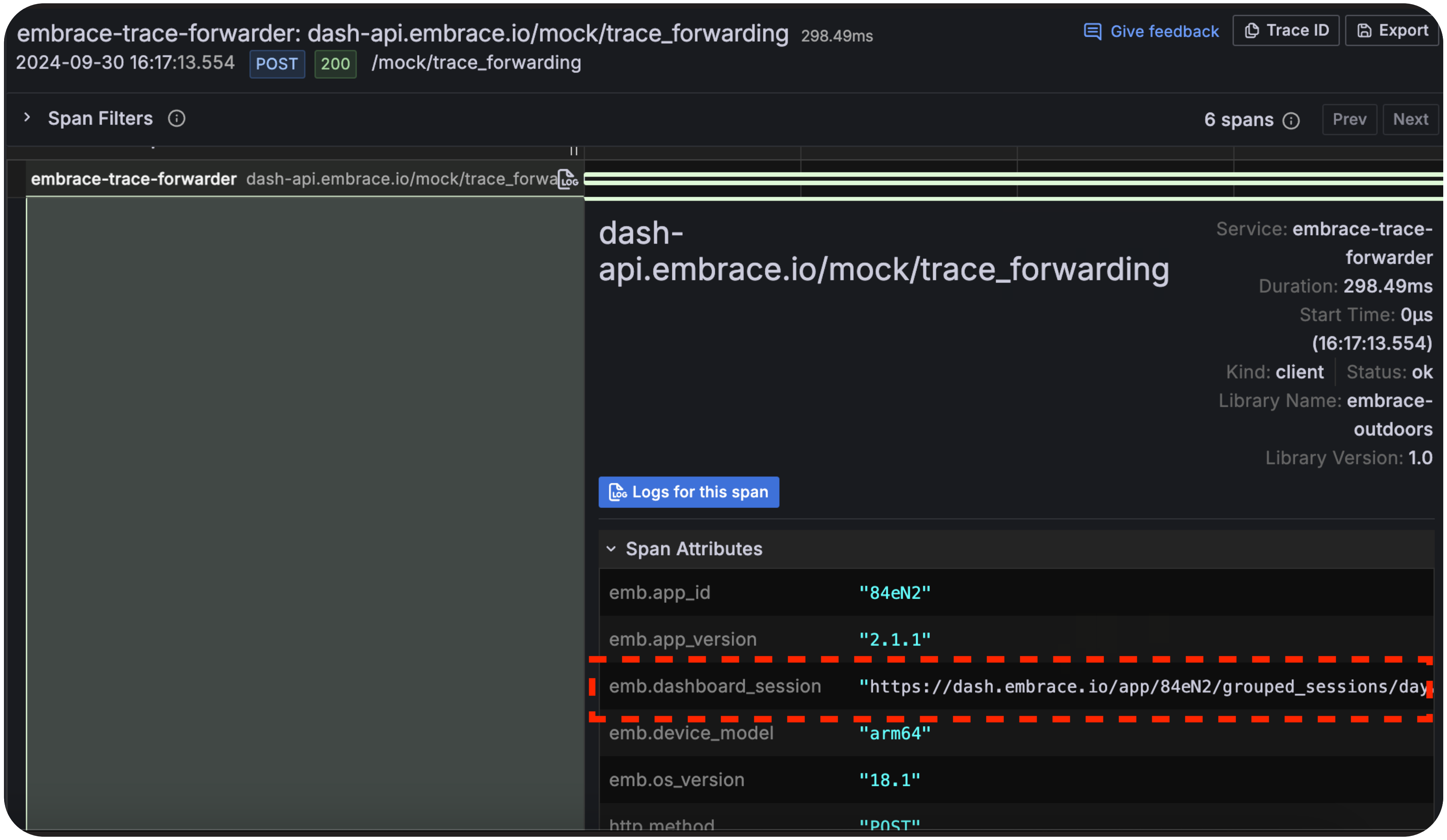


1/23/24, 23:40:19 -> 1/24/24, 00:23:10 | App Version: 10.53.2 | Device: sdk_gphone64_arm64

Start time	Type	Details
Foreground		Session 2 of 3
		Jan 23 23:41:03 213ms 200 0 Third Party
23:41:03	Network	200 OK POST analytics.query.company.com (47ms) analytics.query.company.com/v1/public/yql
23:41:03	Network	200 OK POST server.comet.company.com (73ms) server.comet.company.com/cometd
23:41:03	Network	200 OK POST analytics.query.company.com (45ms) analytics.query.company.com/v1/public/yql

Date	End Time	Duration	Response Code	Bytes In	Bytes Out	Category	Traceparent
Jan 23	23:41:03	73ms	200	309	102	Third Party	00-b0551fe40c80a95471fc1537ddeb7357-7e0d414885d19bf6-01

Embrace also includes a backlink to the individual session where the network request originated, so you can go from your backend observability solution directly to the full details leading up to the network request.


embrace-trace-forwarder: dash-api.embrace.io/mock/trace_forwarding 298.49ms

2024-09-30 16:17:13.554 POST 200 /mock/trace_forwarding

embrace-trace-forwarder dash-api.embrace.io/mock/trace_forwa

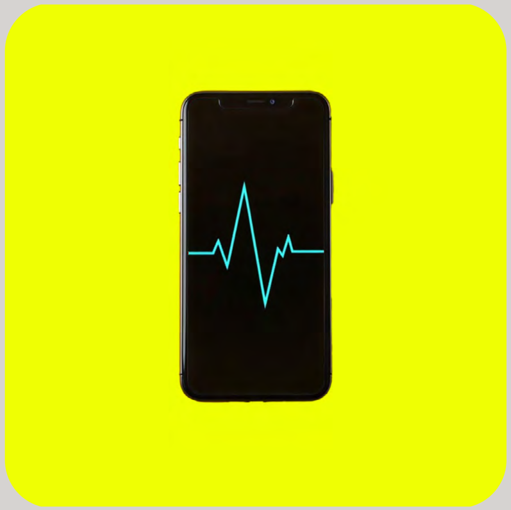
dash-api.embrace.io/mock/trace_forwarding

Service: embrace-trace-forwarder
Duration: 298.49ms
Start Time: 0µs (16:17:13.554)
Kind: client | Status: ok
Library Name: embrace-outdoors
Library Version: 1.0

Span Attributes

emb.app_id	"84eN2"
emb.app_version	"2.1.1"
emb.dashboard_session	"https://dash.embrace.io/app/84eN2/grouped_sessions/day
emb.device_model	"arm64"
emb.os_version	"18.1"
http.method	"POST"

Bringing observability to mobile



While observability tooling and practices have improved significantly in recent years, the lion's share of attention was on backend systems. Companies now realize just how crucial it is to connect the health of their backend systems to the performance and user outcomes within their mobile apps.

Embrace is here to help modernize observability practices for mobile. Our user-focused approach to observability eliminates guesswork by collecting complete user experience datasets. Our investment in OpenTelemetry is improving the open standards of data collection in mobile, so that tools and workflows can have better interoperability. And our innovation in connecting mobile and backend datasets is creating amazing collaboration opportunities between previously siloed teams.

To learn more about how Embrace can help you deliver better mobile user experiences, check out our website or request a demo.

embrace

Embrace provides the only user-focused mobile app observability solution based on OpenTelemetry. By delivering crucial mobile telemetry across DevOps and mobile engineering teams, Embrace illuminates real customer impact, not just server-side impact, to drive success in achieving SLOs. Learn more at embrace.io or follow Embrace on LinkedIn, YouTube, or Twitter.

Contact

1901 Avenue of the Stars #200
Los Angeles, CA 90067

(424)-326-9004

contact@embrace.io

embrace.io