NodeSummit Day One

Lori MacVittie, 2013-03-12

#NodeSummit #LineRate #F5 #devops HTTP error codes, capturing traffic for replay, and APIs for APIs

It was definitely a full day at NodeSummit with panels and presentations on so many different topics that if you couldn't find something of interest, well, you must have been sleeping.

APIs for APIs

Throughout the sessions I attended one theme came through very clear: node.js is exceptionally well suited to building APIs for ... APIs. It isn't that node.js is not a good fit for building APIs in general, just that its asynchronous nature means you can parallelize calls to other APIs, making it a good fit for aggregating data through other APIs whilst minimizing the impact of latency. Which turns out to be a Very Big Deal^{TM} when building applications and APIs for mobile devices. This is also a boon when there is a need to present a RESTful (or more modern JSON-based) API to devices but services are already exposed which present the necessary data in XML. Node.js as an API proxy in such situations enable rapid transformation of data from one format to another without rewriting existing applications or APIs. Go node.js!

HTTP Errors are Transport Errors, Not Application Errors

A second favorite moment was during the "Mobile. Backend. Node." panel session, moderated by OmniTI CEO Theo Schlossnagle. It was he, in fact, who raised the topic of API error codes and blithely announced that HTTP error codes should not be used by application developers.

I'll pause for a moment until you start breathing again. The way Theo explained it was this, "One of the Ts in HTTP stands for ***transport***. HTTP error codes are ***transport*** layer errors, not ***application*** errors." Given that HTTP is the de factor transport layer for applications, I cannot disagree. Granted, I've never really thought about it in this way but once Theo stated it, it seemed obvious.

What we (or We, as an industry) should do about the fact that a lot of applications use HTTP error codes to transport application errors is still unclear. But as GI Joe says, knowing is half the battle.

Capture and Replay Traffic

Last, but certainly not least, was a question raised during our very own session on LineRate. After my counterparts Nojan Moshiri (LineRate Product Manager) and David Adrian (LineRate Systems Engineer) had given their presentation a question was raised from the audience on whether or not one could use LineRate to capture and replay live traffic. The short answer is "of course". When you put the power of programmability in the network with node.js, you can build just about anything you want. Capturing traffic and storing it in some sort of repository and then later on retrieving it and replaying it to applications is certainly something you could build. It's not something we've built or considered building, but it is a great way to leverage the power of programmability in the network via node.js.

It's something folks who practice devops would likely find a good use for, especially in situations where developers are troubleshooting sporadic errors that only seem to occur during live usage but can't justify the investment in a larger, commercial system intended to accomplish essentially the same task.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com