Learn API Basics

This webcast features Dave Slusher, Lex Machina's Developer Advocate, who serves as a liaison between the API product and the developer community that uses it. It will be hosted by Justin Brownstone, principal product manager at Lex Machina and project lead on the development of the API.

During the webcast, Dave provides an overview of API basics: what they are, how they are used, and examples of implementation.

Highlights include:

- Overviewing key API vocabulary
- Live examples of API implementation and usage
- Tips on how to take your API understanding from novice to pro
- Q&A session in which any question is welcomed, whether basic or advanced

Speakers:

Justin Brownstone
Principal Product Manager
Lex Machina

Dave Slusher
Developer Advocate
Lex Machina

Justin Brownstone (00:01):

All right. Welcome, everyone. As a warning, we are recording today's webinar, but that will also be available, along with the slides and a transcript, to answer the first question that's been asked. So, someone already found the Q&A at the bottom. We want a lot of questions today, because we're covering API basics. My name is Justin Brownstone, and I'm a product manager here at Lex Machina.

Justin Brownstone (00:28):

For those of you who aren't familiar with us, I'd like to very briefly introduce it. Now, Lex Machina is legal analytics. And legal analytics help you win. Specifically, we enable you to make data-driven decisions, to answer questions like, "How's a judge likely to behave in a certain scenario?" Or, "What's the best winning strategy for my particular case?" For the last 10 years, we've enabled customers to answer questions like these through our website. But of course, that's all changing with the launch of our new way to access Lex Machina data, through our API.

Justin Brownstone (01:01):

So, that's enough about Lex Machina. In the next 30 minutes, the goal's to learn about API basics. And to do that, we have Dave Slusher here with us today to answer all of your questions. Dave Slusher is a developer advocate with Lex Machina. That role means he serves as a liaison between the API product and the developer community who uses it. He's responsible for creating instructional content, fostering the development community, and representing developer concerns to product management. Dave is also a software developer, science fiction fan, and pioneering podcaster. In fact, when I first met Dave he was being introduced into the Podcasting Hall of Fame. I think he was the fourth podcaster of all time, in history.

Justin Brownstone (01:45):

And he earned a B.S. degree in chemistry at Georgia Tech, and an M.S. in computer science from the University of Louisiana at Lafayette. And previously, he's served as a developer advocate for ServiceNow. So as I said before, we are recording, and also, please, please ask questions using the button at the bottom of your screen. We'll have time for them at the end of today's presentation. And we want to make sure all your questions get answered, from the basic to the technical. So with that, that's it from me, I'm going to turn it over to Dave. Dave, please take it away.

Dave Slusher (02:17):

All right. Thank you very much, Justin. All right, so let me give you the quick rundown of where we're going to go today. We are going to cover the basic question of what is an API. I mean, from what I am told, that is the customer base, there has been a certain fraction of them who say, "Great. Let's get an API. What's an API?" So we're going to cover that. What is an API? Where do they exist? You have a lot of these in your life. You probably don't know them, because they're largely invisible. They're kind of like the power grid. They're making a lot of things happen in your life that if you don't look for them, you may not know they're there.

Dave Slusher (02:53):

What do they do? And then, we're just going to show you the Lex Machina API in action, kind of what it looks like when it does what it does. And then, there will be plenty of time for Q&A at the end. And, just as a housekeeping thing, particularly if you have a timely question about the thing that's on the screen happening, Justin will make the game time decision. He is empowered to jump in whenever, if it makes more sense to ask the question right then. So, do not feel like you have to hold your questions to the end, get them down whenever you feel like.

Dave Slusher (03:29):

All right. So, the very basic question, let's start here. What is an API? An API, at the highest level, is to think as a webpage for robots. It is the same mechanism that serves you a webpage, serves you an API response. It's the same web server, it's the same protocol, it's all the same stuff. It's just a different intention at the other end of it. It is all the data that you get from a webpage, but without the human portions of it, without the styling, and the colors, and the fonts, and things like that.

Dave Slusher (04:07):

Because it is intended for robots, it is in a structured machine-readable format, because it is a machine reading it at the other end of this, largely. I will show you examples of all this stuff, but at its heart, it may sound daunting. An API is this serious, technical thing. It's largely the same thing you're used to. If you're used to the Lex Machina web interface, you will get roughly the same answer to roughly the same question. With a little bit of caveat in that, there's a lot of magic and our data is constantly being refreshed. So, but it is basically the same stuff.

Dave Slusher (04:54):

So, for those familiar with the product, this is what it looks like. I will tell you that every time I do a demo case, I use this same one, it's the only one I have the ID memorized for, because it's about Neil Gaiman and Todd McFarlane, a lawsuit about the ownership of a character from the early issues of Spawn comic books. So, this one is fun to me. But you'll see, this is what you're used to. You load the case, and there's a bunch of information about it. What court was it in? Who were the judges? Who were the magistrate judges? The plaintiffs? The defendants? Et cetera, et cetera.

Dave Slusher (05:29):

This is what it looks like when you load the webpage. And a certain portion of what came down from that web server is the formatting, the styling, what shade of blue am I going to show in the left side? And what color are the links? Et cetera, et cetera. So, a lot of that information is for the human. When you do the API, this is the response for the same case. It's the same title, it's all the same information. It doesn't have all that embellishment. It is strictly for reading of the data. So something that gets this response back says, "What is the case ID?" Well I know that that is in the field called case ID. And here's the title. Here's the court.

Dave Slusher (06:13):

So, when I was talking about the well-structured, machine-readable, this is for our API. This is the structure we use. I'll show you a couple examples of different structures. The one that we use is called JSON, and I will show you examples of that. But, the idea being that, on the left side, that's a human must make this happen. And so, if, for example, this is a thing you do routinely, you're having a human do human things in human time, which can be kind of a drag for the human, and it can be kind of time and possibly money drain. If you're hiring somebody to do something very routine, part of my outlook is, let's have the robots do things robots can do so that the humans can do things that only they can do.

Dave Slusher (07:07):

And so, if you were to look at this periodically, this is really more of a robot job than a human job. And so, let me show you what some examples of these formats look like. So the one that we were just looking at is a thing called JSON. And you don't really need to know a lot about this. I just want to, if you're not familiar with any of these, I just wanted to give you just a flavor of it, so that if you happen to look at that, you just know what you're looking at.

Dave Slusher (07:34):

And it's very much, as you can see, this is very much... It's a fairly terse thing. You say, okay, there's basically a lookup. You say, "Tell me what the users are." It's this list of two different names. And each object has a field called first and a field called last, and there's the names. Now, let's look a the same data represented in a different format. And actually, so out in the world of APIs, JSON is... I don't know the exact-

Dave Slusher (08:03):

Out in the world of APIs, I don't know the exact statistics, Jason might be the most common now. There was a time when this was the most common. It's called XML, and some of you may be familiar with it. You'll notice that it is more rigorous. And in fact, HTML and XML have kind of a common... They're basically they're like apes and chimps; they have kind of a common ancestor on the evolutionary tree. And you'll see that it's very much about having these matching tags, it's very much about the structure. With an XML, you can actually tell if you're missing a field because it's very well defined. Like if you define an address, you know that you have exactly one zip code, you don't have zero and you don't have two, but you may have more than one street address line, but not zero.

Dave Slusher (08:46):

So, you can say when you get an XML, the advantage of it was that you can say, you have two levels of verification. Is this well formatted? Meaning do the tags match. Or is it validated against the schema? Like, do I have all the right amount of stuff in the right place? Now what you'll notice particularly here where we don't have a lot of data, you'll notice that the tags are most of this file. Like if you have a small amount of data, there's actually a lot of overhead. Most of this payload is just the structure of the thing. Now that depends on the ratio of how much data you have, but XML is heavier weight, which is part of the reason why Jason has become the predominant format in practice.

Dave Slusher (09:35):

In theory, XML maybe has some more horsepower, but in practice, Jason is good enough and it's sort of taken over and there's even older, simpler formats, like CSV, which are literally just text files where the values are just listed out and there's comments between them and to this day, there are large amounts of data that move from business to business, exactly like this. And you see that there's not much structure to this thing, so it's extremely simple, it's very human readable. And it's also extremely fragile because if I accidentally put a comma here now between the O and the B, now I've got three fields. And if I put two commas here, I have an empty field for the second one, and this weird extra third one. So it has simplicity on its side, but it's very, very fragile; not at all robust.

Dave Slusher (10:27):

I'm take a quick sip and a quick back breath. Ah. Now where do APIs exist? So, I was telling you that these exist out in the world, and you probably use them every day without knowing it. When you use your mobile apps, many of your mobile apps are getting information and/or doing their actions that's how they do their work. These kind of restful APIs exist out on central servers. You open your Twitter app and you are actually speaking to the Twitter API. You're not loading a webpage, you're not doing anything like that; you're actually using the same kinds of APIs to do your work, to read and write.

Dave Slusher (11:12):

Automated processes. And particularly here, I'm thinking of like enterprise to enterprise, big B2B processes. In my former job, one of the fun programs I did read an API that gives flight information, flight status information. So every time we booked a flight, we would add this to that, and we would be

able to track where our flights were, but Google Flights uses that same API. There's one provider of this information, so if you are looking at anything that gives you real time information of a flight status, you're looking at that same API, serving that information to a different automated process.

Dave Slusher (11:51):

And smart devices. I made the mistake in our practice session of using the name, I'm not going to say it because everything in my office will light up. But if you use any of the well known home assistants that activate my voice and start your Roomba, you are a using an API. There is an API that says, "Start my thing with this, to clean this room or clean everything." And those are driven by APIs. So let me show you an example of this, of mobile apps. This was a couple days ago, this was what the weather was. This is a screenshot of my phone looking at the weather, but what it's doing when it does that is, it's actually going out and asking for the information and it gets it... This is not the exact thing, but it's a equivalent API. So this is, you're going to get information of something like this.

Dave Slusher (12:42):

And much like the example where I was showing with the webpage, you can see that this is all business. This is all the data. You can even see the temperature, I believe it's in Kelvin. So it's not showing you the temperature in Kelvin; your app is doing the translation, figuring that out, but it knows exactly what all the information is, and then it chooses how to display it to you. But underneath the hood, when you open, in my case, this is WeatherBug, this is how information moves back and forth.

Dave Slusher (13:12):

So what do APIs do? Like I said, you have basically read type information and the Lex Machina API is this, right? You're not writing data to our database, you are reading data. And some, and again, with that flight API that I mentioned, that's largely a read API. There are right APIs that I mentioned the Twitter API. Like when you send a tweet, you are pushing data out to Twitter, which will then become a tweet in their database and notified to the world, right? So it can also accept data and it can affect changes, right? This is the smart home example, but also industrial things. You can have control of various power, for the big enterprise control and command things, you can affect those changes out in the world, right? Turn the power plant on, things like that. These are one would hope, highly secure, but these sorts of things exist out in the world.

Dave Slusher (14:24):

These are typically structured. I'm going to tell you this, not because you need to know this, just so that you happen to run across the terminology, you at least have seen it, and it's a little bit familiar. There's five HTTP verbs; there's GET, which is when you load any standard API or any standard webpage in your web browser, that's what you're doing. You've done a GET.

Dave Slusher (14:45):

You got a POST which pushes information back that you are now acting on the other side of this. So if you are submitting a web form and you hit that submit button, you are doing a POST. And sometimes, some little process might be doing a POST, transparently under the hood for you. But the vast majority, when you're using a web browser, you're doing these two 99.99% of the time. There exists three others; PATCH, PUT and DELETE, which are really just for writing. So like your Twitter, maybe not Twitter, but something that's writing to a remote database might be using PATCH, PUT and DELETE in practice in your web browser. You probably have never done this, but they do exist. Not in our API, because again, it's not a writeable API, but these things do exist. And you will see when I demo the API, you'll see that, presently, we have one POST and everything else is GET because you're almost always reading the data.

Dave Slusher (15:53):

So here's the kinds of things, this is not at all an exhaustive list, but these are the kinds of things. One can use an API for in business. One of them-

Dave Slusher (16:03):

.... things one can use an API for in business. One of them might be that you would hook it to a visualization tool. There exists a big amount of raw data out in a remote database, and what you want to do is see what fraction of my cases are open right now versus closed. Or how many have a trial date in the next two weeks? Or something like that. You could conceivably hook the API up and then send it to something, to a process running either in your intranet or somewhere that's assembling this into a dashboard. So you basically have that single pane of glass. So that instead of it being a bunch of different searches and then cutting and pasting data, or somehow manipulating the data, you have an automated process, that's pulling Lex Machina data down and then presenting it in the form that you want.

Dave Slusher (16:54):

Conceivably, you might want to do some form of data analysis, where you want to have it locally, whether that's in a spreadsheet or you have some sort of analysis tool. You might think if you have some machine learning type thing, if you got some serious AI. You say I want to look at every case in California and run it through this AI to see some analysis, you could do that sort of thing. Merging data sources. If you have a CRM, you have a database or a tool for managing your customers, it is quite possible that you maybe go out, you have somebody who goes out and looks at Lex Machina at some regularity to say is this customer facing any new litigation that we don't know about?

Dave Slusher (17:50):

With the API you don't need a person to do that, you can just have that done again by the robots. That sounds like a robot job, let's have a robot do that. For things like that the analogy that I use is it's one thing you can always back up your computer anytime, right? You can hit a button and back up your computer. But it's maybe more appropriate to plug into a time machine and have that do that for you all day, every day, and then you don't have to remember to do it, it is a thing that just becomes part of the regularity of your business processes. You can drive your business logic. You can get the data from Lex Machina and you can then use that to make decisions.

Dave Slusher (18:36):

However that feeds into your decision-making process you can use that raw data that way, and automated workflows. One of the examples, I was running with Justin some examples and one possibility might be you're about to hire a new litigator. So maybe as part of the interview process, you have a process that just goes out and looks up every case that this person has been involved with and bring that data down and present a packet, so that the person who's interviewing knows how to do that. Or you could conceivably do the same thing about all the opposing counsel for every case that you're on. Get me all the intel that I need about everybody else in here. All right. So with that, I'm going to head to the demo section. While I'm doing that Justin, if there's any questions that you want to, that have come in that you want to address while we're...

Justin Brownstone (19:30):

A quick one that I thought was great was how secure are APIs and are firewall configurations needed?

Dave Slusher (19:36):

You don't need a firewall configuration to consume it. I'll even show you since I've got this right here. This is a tool called Postman, which is really just for sending these requests. What we have is a thing called OAuth client credential flow. What that means is you go to our site and you create some credentials, and those credentials become part of the authentication mechanism. I actually have in here a token that's part of this, you'll see part of the secret, not enough to do, it's truncated so you can't use my secret. But you get an access token, which is good for 60 minutes. Then that access token is what is required to access the API. I actually refreshed it just let me, I can do it again. I refreshed it just here a few minutes ago, so it wouldn't run out, but you have this token.

Dave Slusher (20:36):

If you're joining on the spot you could actually authenticate with my token. Those are good for 60 minutes and that's how we control access to the API itself. Let me show you some, let me start with just some of the listing things. For example, if I come over here and it might get an error the first time because I've got the new... You see right here, this is the list courts, and this one is showing every court that we have in our database. Again, in that same structured thing. District court of New Jersey, of Columbia, of Oregon, etcetera, etcetera. We have a similar type where you were listing the case types, and these are all the types. When I hit that button, basically I went out and I said go to apilexmachina.com and do this function, which is list case types. What you get back is this thing.

Dave Slusher (21:33):

These are largely the same from day to day, but not always. This can add over time. For example, this might be a thing that once a month you just go and check and say is there a new case type I don't know about? You could, for example have some notification mechanism. Say we are going to check once a month and if there's something on this list we don't know about, I want to know that, that exists. You can do whatever kind of functionality like that, that you want. I had mentioned before that there is only one post that we have right now. When you query you send basically this fairly, it may look a little bit complicated, it's a little bit complicated because it's a complicated thing that we're doing. Our query has a lot of moving pieces to it.

Dave Slusher (22:28):

I'm sending the body of this post is in fact the query that I'm sending to the server. I want to know cases that are terminated, where the case type includes in their contracts. It was filed after 2008, before 2010 and terminated 2015 in the Court of New Jersey with a million dollars in damages. I undermined my own thing by having this already there. But there you go, those are the two cases that came back from that specific query. You'll see that what we get back from that is basically the raw case ID rather than the whole list of information about it. So then what you do is you go back and you have another, you have this other endpoint, that's the terminology for these things, these are endpoints.

Dave Slusher (23:22):

Let me send the request for that and I will get back the information about this case. It's looking it up from the database and you see that this is the case, Schiavone versus Dragados in case of New Jersey etcetera, etcetera, filed on wherever, filed in 2009 terminated in 2016. So you see that this is particularly when you're querying, this is kind of a multi-step process. Partly for performance reasons, and because these packets can be big so you'll notice that this is a pretty substantial packet right here. The query returns those lists of cases and then you can one by one look up those cases.

Dave Slusher (24:02):

... and then you can one by one look up those cases. So that's a very quick overview of that. Let me show you really quickly an example of using these things out in the world. So I came to Lex Machina from a

company called ServiceNow, and we're looking at ServiceNow, right now, and hopefully all my stuff has still stayed alive. So this is a thing called their virtual agent, and I've got some various accesses to our API setup on this. So it's asking me for a case ID. I'm going to give it my favorite case. You'll see a little bit of debugging message in here because it's running in authoring mode, but you see that you get back this thing. So that same case I showed you before, Neil Gaiman versus Todd McFarlane, et cetera, et cetera and some information about the case.

Dave Slusher (24:52):

It's a copyright case. It went to trial, appealed. It was contested. It went to a jury trial. Let me show you under the hood. This is what happened when I just did that is it executed this workflow that down here actually went out and did this transaction. It says, "Send me the same thing that I just did with this Postman tool, so send me this thing with this case ID." And it returns back this, and you'll notice that this is lots and lots of information, and I didn't present all of it. I presented a fairly small subset of it over here, because I have a formatting step over here.

Dave Slusher (25:33):

Again, this is a very simple example, but you can do effectively, anything that you want in here. As I have a step where I say, "Show me the title. It was filed on the date. The status is this, et cetera, et cetera." So this could be anything that exists in this case, but if I gave it a different ID, then I would get a different answer for all these things. That's like a concrete example of an API actually doing stuff. All right, and with that, let's turn it over to Q&A.

Justin Brownstone (26:05):

David attendee wants to know why that's your favorite case? It came up a couple times.

Dave Slusher (26:09):

Because I'm a comic book fan. I listen to a podcast where they do responsive readings of court transcripts, and depositions, and they covered all the depositions of that case, where one guy reads Neil Gaiman, and then when somebody reads all the lawyers. That sounds a little dry. It's the most fun thing. It's hilarious.

Justin Brownstone (26:32):

And someone [inaudible 00:26:33] at the follow-up, which podcast was that?

Dave Slusher (26:35):

It's called Cartoonist Kayfabe. This is absolutely not the question I thought we were getting.

Justin Brownstone (26:40):

No, I think it was about ownership of a spawn character, correct?

Dave Slusher (26:44):

Yeah. Correct. The character, Angela. Yeah.

Justin Brownstone (26:46):

We got a couple questions about selling the API, purchasing the API. I would say those questions are better directed toward the sales team or your CSM. If you're already a Lex Machina subscriber, you could also reach out to me directly. I'm happy to facilitate getting you in front of the right person. The plan is that it will require enterprise access to have access to the Lex Machina API. But again, I'm probably not the best person to answer that question. So if you have questions about getting access to using it, then please reach out.

Justin Brownstone (27:18):

I will say that Postman, which you just saw Dave using is incredibly easy to use. You don't have to be a programmer. Dave has written a description of how to use it on our website. That link is going to be on the next slide. I went through, and was able to do it. I actually used it to access a different API, not the Lex Machina API. So if you want to play around with anything you just saw Dave do today, he's got a detailed write-up for you on our site.

Dave Slusher (27:47):

We took the slide up, but it will be in the follow-up email. We'll have a list of all these resources.

Justin Brownstone (27:51):

Okay.

Dave Slusher (27:53):

Yeah. It's actually fun. Typically, When I'm learning a new API, typically, I fire it up in postman, just so I get a sense of what things look like. I don't know, more of a plotter as a programmer, right? It helps me to actually have the, a concrete example of that payload, and then I start programming to that thing, typically.

Justin Brownstone (28:14):

Here's another interesting one, user rights, I have the ability to bypass APIs in my security software. Why would this be needed?

Dave Slusher (28:25):

I'm not sure, I understand that question. Bypass API as in security software?

Justin Brownstone (28:29):

Okay.

Dave Slusher (28:29):

They're saying filter APIs out or allow them?

Justin Brownstone (28:36):

Yeah. I don't know something, you know, but Tina, Lex Machina user, send us an email, and we'll figure it out together. We can follow-up in more detail later today or soon, allow, we'll do what she says. [inaudible 00:28:47]-

Dave Slusher (28:46):

That is intriguing. I'm not sure off the top of my head, what that context is [inaudible 00:28:52]-

Justin Brownstone (28:51):

All right. Yeah. Hopefully, Tina will reach out, and we'll get that. Can you use the Lex Machina API to compare data, for example, parties to your client list? I think this goes to what you were talking about earlier.

Dave Slusher (29:04):

Yes. You absolutely could, right? That's If you've got a list of prospective clients, you can look and see what litigation are they in. So that is exactly the kind of the use case that this is good for, right? Is effectively you can think about the Lex Machina API as a data pipe. So, I kind of showed a few concrete examples of that, but think about anything you can use that data for once you have access to the API that enables you to do that, right? One of the beautiful things about these is... and the thing I'm looking forward to is the first use of our API in a way I would never have even conceptualized, right? There's always how customers use the APIs, is that they do something you would never have dreamed and you never would have built for yourself. That's part of the fascinating thing is that once you have the data, whatever it is you think you need, you can just do.

Justin Brownstone (30:04):

Someone asked, are there modules available for Python to read API data directly?

Dave Slusher (30:10):

Not yet. That is on a roadmap. That is a thing that I would like to do is provide for some of the common uses, like a Node. So basically a pre-configured say Node package or Python package that already knows that structure that we were just showing, and so that instead of you having to parse it directly, you get a thing called Lex Machina and a case dot case ID tells you what you want to know. That does not yet exist. I have not been at this job long. That is on my groaning and always expanding to-do list. So we want that to come out. I can't make a promise on timing, but that is a thing that will exist at some point.

Justin Brownstone (30:53):

Great. Francis, sorry to pull you off of mute here, but I'm a bit new to these and I notice we're over time. We still have questions outstanding. I think maybe what we can do is ask those of you have outstanding questions. Do we have a contact here that they can email and we can get them answers to their questions on this last slide?

Francis Garcia (31:16):

What they can do is email Dave directly on the last slide, and then Dave can get back to them in regards to the questions they do have for him. Since they are specific questions and you can assist Dave with answering those questions as well as he gets those emails.

Justin Brownstone (31:31):

Great. All right. Well, thank you so much for everyone for attending. I'm sorry, we ran out of time. But like I said, anyone who has outstanding questions, dslusher@lexmachina.com. I'm jaybrownstone@lexmachina.com. So please just copy and paste that into an email, and we'll get your question answered. Thank you so much, everyone for the time today.