Lex Machina®

# API 2.0 Webcast: From Basics to Better Decision Making

Lex Machina is proud to host the second webcast in our series on leveraging our API. In the first webcast, Learn API Basics, of our series first recorded on May 17, 2022, Dave Slusher (Lex Machina's Developer Advocate) outlined API basics for attendees interested in this crucial topic. In this next webcast, Dave will show how APIs go from abstract academic pursuits to enabling new features that are impossible without the API. At the end of the webcast, attendees should have a concrete understanding of some ways in which they can utilize APIs to improve their businesses.

## Speakers:

Justin Brownstone
Principal Product Manager
Lex Machina

Dave Slusher
Developer Advocate
Lex Machina

Justin Brownstone (00:00):

Welcome to the Lex Machina Webcast, API Basics, The Sequel. My name is Justin Brownstone and I'm the product lead for our API. Now, API is a big buzzword in legal tech right now, which is why in our last webcast, we broke down the basics. What an API is, what it does, how they are used. But as Dave Slusher will say in his presentation today, now what. Hopefully, by the end of today's webcast, you'll have a better idea of how to think about utilizing APIs. Today, we'll specifically be talking about the Lex Machina API, but I'll warn you this is not a product demo.

We're not going to go in depth on all its different features. Instead, Dave is going to use our API to demonstrate how you go from thinking about business problems you want to solve to using an API to solve those problems. Please, please ask questions. Use the QA button in Zoom at the bottom of your screen. We'll have time or we'll try to have time for them at the end. If we don't, we'll make sure they get answered. The purpose of today's presentation is to answer those basic to technical questions. Now let me introduce Dave before I hand it over to him.

Dave Slusher is a developer advocate with Lex Machina. He serves as a liaison between the API product and the developer community that uses it. He's responsible for creating instructional content, fostering the development community and representing developer concerns to product management. Dave is also an experienced software developer, science fiction fan and pioneering podcaster. In fact, as we were just discussing before the call, Dave was a 2022 inductee into the Podcast Hall of Fame alongside Marc Maron, so pretty cool. With that intro, Dave, I will turn it over to you.

Dave Slusher (01:48):

All right. Thank you very much, Justin. So let me kind of run over what we're going to talk about today. And unlike the first one, I'm desperately trying to go short so we get more time for questions this time. So first, I'm going to do a lightning recap of what we talked about the first time. Then I'm going to talk specifically about our API. So we kind of studiously avoided that in the first webcast kind of trying to stay above the frame, but now we're going to talk about what we provide in our API. And then we're going to talk about the fact that that API exists, you have it, or you're thinking about getting it.

What happens next? That's a kind of critical point in this life cycle. We'll talk about some of the use cases for using the Lex Machina API and then we'll talk about how you think about the decisions you want to make and tie that to what the API can do for you. Because ultimately, this whole thing is not an academic concern. If this doesn't provide business value, then why are we here? That's what we want to do is provide business value. Then I'll do a quick demo of one way that it can be used, one of many ways, but one concrete way. And then we'll take some questions.

So lightning overview and I'm going to condense my previous half hour into two slides. So what is an API? An API is a webpage for robots. Take all the things that you get from an API, subtract all the formatting and leave behind just the data in a structured deterministic machine readable format, so that a process that's pulling this down knows exactly where to look in there for the data. And as an example, I'm going to look at the same case two different ways. For those who are Lex Machina customers and familiar with what we do, this is a familiar page to you.

But here is the identical case as served out by the API. You'll see, it's got the same title. It's got the same participants in the same court, but this is a format known as Jason. And you can use this to know that if you're looking for the field named Case capital I ID, you will find the case ID and all of these things are understood, consistent and very automatable by a machine. So the Lex Machina API. An API product has four main kinds of endpoints. And when I say endpoint, that's just the EARL, the web address of a thing in the API is an endpoint.

So we have four kinds of these things. Those are lists of resources. And so we have, and these pretty much correspond to things you can filter on in the web UI. So you can get a list of every case type that is a valid one in our system or tags or case resolutions or things effectively that you would use as criteria to limit what you're looking for. We have a looking up of an entity by IDs and entity really is kind of a weird computer word that just means a noun, a thing. This could be a case. It could be an attorney. Like all the people are some form of entity, either they're a party or an attorney.

A law firm is an entity. A judge is an entity. Searching for entities by string. So we now have the ability given a text like a name or a part of a name, we can search for judges, attorneys, parties, law firms. And then the result of that is to get back that ID that you can use for other things in the endpoints above and below. And the one that really drives everything. The key part of the whole deal is the ability to query. So given some constraints based on these things that we have, the previous items, given some constraints, give me the list of cases that match those constraints.

These can be date constraints, type constraints, participant constraints. Show me every case filed in the last five years against a particular judge in a particular court, whatever. You have very nearly infinite ways you can construct these constraints. And then you get effectively a list of case IDs. And going back up to here, you can then look these cases up one by one. That's kind of the pattern of how you use these. All right. So now the big question is now what? So, that thing exists. What do you as a Lex Machina customer do given that?

Let's say that somebody told you, you need the API and you remain to be convinced by that. And you're saying, well, what does it do for me? How do I think about this? So the first step of that is basically the slide that I just showed you. Understand that. These are the things that the API can do. These are the capabilities that it can provide. So kind of reading between the lines, if there was something you needed that's not on the list, it's not a current capability. But you can do practically everything you do with the website with those four kinds of endpoints.

Then look at your current use of the website. So you pay for the website, you use the website, how is it that you use it? What are you doing with it? How often are you using it? Kind of understand what it is that you're using it, like who's using it, what they're doing, how they're getting value out of it. Think about whether any of that usage that you're currently using the website for can be switched over to the API. If you've got somebody periodically cutting and pasting from the website, that is a prime indicator that maybe this is something that could be done via the API.

Dave Slusher (08:00):

... Indicator that maybe this is something that could be done via the API. But, where it really gets interesting is, think about, now that you kind of know what can be done with that. Think about what things that are difficult or possibly impossible to do with the website that you could do with the API, but you've never done because you couldn't, but you would like to. What are new ways that you could look for information that are enabled by this? That is, like I say, net-new. These are hard to think about, because it's much easier to think about what you already do, then what you don't do. But, if you had something that was on your wishlist. Like if you have asked Lex Machina to add functionality to the website, that's because you have a need that's not presently being fulfilled. Think about, is that need something that you could do by the API? There's a very good probability that the answer to that is yes.

As I said before, none of this is an academic concern. We have an end goal, which is for you to produce more value to your business in less time. And less time, more or less directly corresponds to less money. So let's talk about some use cases for the Lex Machina API. And this kind of overlaps with what I was talking about before, or it's in some ways a refinement of that. Periodic retrieval of data. As I said, if somebody in your office does a weekly summary on the website and then cuts and pastes it into a report, that's probably something that could be done completely via the API, without a need for somebody to do that over and over again. All right. If you are generating summary and statistical data, that is exactly the kind of stuff that this is really good for.

If you've got somebody who part of their job is to go out and periodically do a search, looking for certain states of data, certain events or conditions or something in the data, has a new case been filed in this court or against this participant? These are the sorts of things that are very good API use cases. These can be set up and done. The periodic retrieval it kind of implies, you're not the one doing it. You've got a robot that's doing this on a timer in a background. And what it also implies, if there is no change, I don't even want to know there is no change, just tell me. Send me an email or however you hook it up. A Slack message when something changes or something we as a business need to know about, and then just quietly sit and do your robotic job until then.

And then, this is a similar concept, automating repetitive tasks. In my intro, Justin mentioned I'm a science fiction fan. Science fiction people of a certain age got a lot of Asimov's Rules of Robotics. The Laws of Robotics, they're all about obedience and protection. But there's a newer rule of robotic you can think about, which is, if a robot can do a task, a robot should do a task. Because when they're doing those tasks, that's freeing up the people to do the things robots can't do. So if you've got really repetitive usage of Lex Machina, generating data, have the robots do that. And then you can do a higher level analysis, looking for patterns and looking for value that a robot could not do.

And then, what we were talking about before is, you've got all of this, you've understood the API, you have done everything that we've asked up to this point. Now, what are you going to do from this point forward? And this is where you begin to think, again, to stress, none of this is academic. This needs to be accelerating and refining how you do your job. Making it better, making it cheaper, making it all of the above. So now the question is, what decisions do you make in the course of pursuing your business that you would make differently if you had more information? If the cost of obtaining that information goes down dramatically, the time and cost, which are very closely related, if that goes down so that you can get all the information you really need, what decisions would you make? Would you allocate your resources the same if you knew more about the current state of litigation? Would you prioritize work the same if you knew more? Right.

You can think about it, like what would I do if I were unlimited by the ability to gather information? What business value could be improved if we had access to that information? Whether it's acquiring new clients, whether it's being of better service to existing clients, if you are a Lex Machina customer who is not in direct representation, could you plan better for the future, knowing more about the current state of litigation? Think about the fire hose of data. If you can turn that fire hose on, what will you point it at? What will you do, what will you think about? When you know the questions, now it's a matter, it's a very simple, almost mechanical matter of going back to the API and saying, boy, if we knew these things, we could really plan out our next year better.

How do I get that information from the API? And once you know the question you're asking, it is a fairly straightforward matter of achieving that. I'd like to know all the cases that happen in this court with this particular set of metadata. And if I knew that, then I could do a better job. That's the way that you think about that. All right.

So, with that, I did pretty close to keeping to where I wanted to on the presentation. I'm going to take a quick sip of beverage. Is there anything Justin, that you want to mention before we go to the next phase? Anything jump out.

Justin Brownstone (14:41):

There's a couple questions that I wasn't quite sure I understood. So, hopefully that person can either add a little more detail or feel free to reach out to us. Or anyone who's not comfortable posing a public question, reach out to JBrownstone@LexMachina.com or DSlusher@ LexMachina.com afterwards and we'll make sure to answer your questions. But yeah, I'm looking forward to the demo.

Dave Slusher (15:07):

All right. So, what I'm going to do now is I'm going to cut over to show you the API in action. This is one way of getting data, this is far from the only way, but this is one concrete way. And what I'm doing here is I have a script that's running a node on my local machine. This is a way basically of writing JavaScript code and creating an application in that. Just yesterday, we published a package to the node package manager that is an interface with our API. So this thing that we're looking at here, the thing that powers it, this Lex Machina client object here, you can get this stuff right now, bearing in mind that you have to be an API customer or have access to the API. So the whole world has access, but it's a little bit of a mood-

Dave Slusher (16:00):

But it's kind of a little bit of a moot point if you don't actually have access to it. But let me show you what this does. This is a very simple script here, where we create this client object and a query object. And we get a law firm, which in this case is the California Department of justice, and a number of days. Let me go ahead and actually start running this while we're doing this, because it will take about 30 seconds. You can see that this script now found 47 cases. It's looping over that list. I'm having it output the title, just so you can tell something's happening and see what's going on. We're going to see my alma mater coming up in a second here, I think.

So what we do is, this line here is really just about ... Oh, there it is. University of Louisiana at Lafayette. You see, this really is just about getting a date, 60 days ago, in the right format. We use that as a set date. We add our law firm, California Department of Justice. We query district cases. This page-through just means, if there's more than one page of data, just keep getting them all until you run out of data. And then we get the list and we iterate over it. You'll see that after it did that, it got its 45 cases and the most common types. Remaining federal. These are just types that are in there. These are the tags. I'll be honest, probably every person on the call knows more about what they're looking at than me about this stuff. But the meaning of these things, but how you get it ...

And then I just did a little bit of, I looked to see the law firm. Who are the represented parties? And I looked to see whether the damages were for those parties or against those parties. So, this kind like of a little scoreboard. How much did we win and how much did we lose? This was actually pretty small for the Department of Justice. If you look at some of the private law firms, it can be a lot more than this.

But that's what it does, and that's just one way of doing this. But whatever this is ... When I was talking before about knowing what question you want answered, this is the question. The date is 60 days ago. The law firm is California Department of Justice. But there are billions of questions, because you have about 35 different criteria, or maybe even more, that you can add. And given those criteria, you can really refine that list into exactly what the answer is that's meaningful for your organization.

So, this is fun. I'm a programmer. I could fiddle with this ... Literally, this script, I could spend three weeks on just fiddling with this. There's a similar script where we look to see, given a law firm, which attorneys have faced it, and it goes out and does a search by the judge. And you can always add more fun stuff like that, and that's kind of the stuff I like to do.

So anyway, with that, that gives us a good bit of time for Q&A. So, let us go to ... Oh, well, one more thing before we go to the Q&A. As I said before, that Node client, it exists for those who kind of understand ... If you don't know what this means, don't worry about it. But for those who kind of understand Node, this exists right now. You can do an NPM install, and you will get the Lex Machina client. And if you hunt in NPM, the website, if you hunt for Lex Machina, you will find this and our information. So good reason to have the API, because then you can fiddle around with this. All right. With that, we have allowed, I think, sufficient time for Q&A. So, let us go to it.

Justin Brownstone (19:42):

Great. Yeah. Thanks, Dave. I'll say also, I'm not a programmer and I've fiddled around with that package too. And it's pretty easy to plug in different law firms, different date periods. It's really fun. Dave's written documentation on how to get started using the API without being a programmer. So certainly, anyone can start playing with this and understand what they're doing. So one of the questions I have here, Dave, is, "Can you integrate the API with SharePoint team sites?" A lot of questions we get are about integration. Maybe you can talk about how APIs are used that way.

Dave Slusher (20:17):

So I don't know the specifics, and probably most of the specifics that you asked me, I probably won't ... I don't know anything about how SharePoint consumes endpoints. What I can say is, anything that will consume a RESTful ... And it is capital R-E-S-T. REST is a term of art in this. Anything that can use a REST endpoint can use Lex Machina. So, I don't know. Off the top of my head, I'd have to do a little research. And I certainly can, and we can get back to people on any specific like this. Again, email me, dslusher@lexmachina.com, and I can look and see. Off the top of my head, I don't know what SharePoint does. But if it has the capability for consuming a RESTful endpoint, we are a RESTful endpoint. So, I'm going to say a qualified probably. I mean, the answer to, "Can we integrate X with Lex Machina?" The answer to every one of those is a strong probably.

Justin Brownstone (21:21):

Yeah. This a kind of similar question, but I thought really ... Thinking along the lines of questions you've raised, someone said, "Can we create a series of search forms on our internet that would pull data from Lex Machina? Some of our users find it overwhelming. Could we create a form where the attorney fills in or picks criteria for a common query, sort of like LM apps? It might make Lex Machina less daunting."

Dave Slusher (21:45):

Absolutely. So, you totally could do that. With some very slight modification, you could use this thing that I just showed you as basically the data-driver layer. So you could have a very simple website, where you basically want to hide a lot of the complexity. That thing I was showing you actually hides a lot of complexity. You configure some stuff about it, but you are not worrying about an access token and a lot of the fiddly bits about it. You just ask a question and get the answer. And you want to basically do a similar level of thing, where you want to hide some of the Lex Machina complexity. So maybe it has two or three fields on it, and then it does basically that same sort of thing. This is totally doable.

So, it is absolutely physically possible. I don't, off the top of my head, know how hard it is. I don't think it would be a huge development effort. I mean, one way or another. There might be simpler ways than using our Node package, because like I say, these are just endpoints out in the world. So anything that can call AJAX, which is just a way of asynchronously doing website calls from a webpage. Anything that can call AJAX can also use our website or our API. So you could easily build an angular, a React App, that calls this. It would probably be pretty simple, really. Any developer who can write a React App could probably integrate with this pretty easily.

Justin Brownstone (23:27):

Absolutely. Yeah. One of the questions that keeps coming up that I'll answer is, "Are you able to pull documents, or dockets, information from dockets?" The answer to that is not yet, but it's on our roadmap for 2023. So absolutely, within the year, we'll have dockets and documents available via the API as well. "Is there a timeline on additional states being added?" Might be a question for another ... So I'm not the expert on the state timeline, although right now the API covers district court cases.

Justin Brownstone (24:00):

Although right now, the API covers district court cases, and appeals data and state data are next on our roadmap to be added to the API either end of year, this year, or Q1 2023 as well. There's a lot of questions about matching parties, linking parties to cases. That may be a single person who should reach out to us so we can go through this together.

(24:30):

But, Dave, I wonder if it's worth talking through a little bit how party IDs are used in this system, and how those are used in this search first match on the case Jay's on.

Dave Slusher (24:43):

Sure. Do you want to talk a little about the centroids, or is that too much?

Justin Brownstone (24:47):

I think that's too much. Yeah.

Dave Slusher (24:49):

Okay.

So effectively we have, for any physical person out in the world or any physical organization, we have basically a canonical ID that represents them. And so that is the ID that I was mentioning before. So when you look at that case information, for example, as you go down in there, there will be a list of, say, law firms. And it'll be law firms, plaintiff law firms, and there'll be one or more. And it'll be a list of those IDs.

And so when you look at a case, you will basically have lists of those IDs, whether it's for a person, for the plaintiff or the defendant or a third party, anything like that. Those are the same IDs that are returned if you do these search. So all of these things interact with each other, right? You can do a textual search for a person. As in any searching context, searching for John Smith is going to be harder than searching for Chip Zdarsky and getting the person that you mean.

So our API doesn't solve that underlying problem. But given that you can find the person you're looking for, then you can find their ID. Given their ID, you can use that ID in a query. You can use that ID to look that entity up, whether whichever it is. It could be a patent. It could be a party. It could be a law firm. It could be a case. These are all things that can be looked up by that ID.

I'm not sure if that answered the question.

Justin Brownstone (26:28):

I think that's helpful. And there's also more information about how IDs work in our documentation. Or like I said, if you have more questions, you can follow up with DSlusher@LexMachina.com or Jay Brownstone. Just first initial-

Dave Slusher (26:41):

Let me go ahead and do that.

Justin Brownstone (26:42):

There we go.

And please, please, don't hesitate to reach out. Dave, loves talking over these things.

Let's see. Is it on the roadmap to search for a company and get the parent in all the subsidiaries? The answer is yes. That's the other project I'm working on. So I actually selflessly chose to answer that question. We are tentatively, hoping to release industry tagging as part of that before the end of the year, and then parent and subsidiary relationships added to the database or early next year.

We'll take this as the last question. So theoretically, I could plug Lex Machina into our client database and generate a persistent correlation between our client numbers and Lex Machina party IDs.

Dave Slusher (27:31):

Absolutely. Yes. And this is one of those things that might be good to be a semi-automated process. You could totally go through your, basically list through whatever it is that's currently holding your client information if you have some CRM or something. Go through name-by-name and search for them. And obviously, some of those names will generate more than one hit. And so that might be a thing where you go through, and somebody might have to say, "Oh, this is the one."

But at that point, you can persistently say this CRM ID is the same person as this Lex Machina ID. And so now you have a really good pipeline from your CRM, and now you can say, "CRM, once a week, tell me what's up with this client. Have new actions been filed against this client in my CRM?" Well, who are they in Lex Machina? Now let's go search for that in Lex Machina. So once you have that, that's actually a really valuable thing. So the creating of it, because of the nature of that, because you are lexically matching people out in the world who may have greater and lesser uniqueness to their name, but once you have established that basically lives forever and is an extremely powerful set of information that you have.

Justin Brownstone (28:59):

Awesome. Thanks, Dave.

All right. Well, we hit the half hour, so we'll stop there. Thanks, everyone, who attended. Again, we'll have a recording of this available if you're interested. And please follow up with an email to Dave and/or myself, if you have any further questions. Thank you.