Krish (products.snowpal.com) (00:00.974)
Hey there, welcome to Snowpal Software Development and Architecture podcast. In this video, we're going to take a look at one of Snowpal APIs, the custom attribution API. As you probably know, one of the products that we have is a suite of APIs. I call it one product because it's one category of products, but it's a number of APIs with thousands of endpoints. In this video, we'll just talk about one of those APIs. And just take an example.

talk about how you might have to build something if you did not have access to those APIs, to that API, sorry, and how much time, effort, money, and risk you could actually potentially save if you did integrate the custom attribution API and had the ground running. So without further ado, let me share my actual screen here.

Okay, so you can go to custom attribution API dot snow pal dot com with custom attribution API being dash delimited. It's typically the way our postman collection DNS custom domain URL works. You can also go check out our developer documentation at developers dot snow pal dot com. We've seen the product managers, product owners like to start with developer documentation here. Sometimes the developers want to just skip and write, you know, go straight to postman.

because they are, sometimes some developers want to just discover the end points as they go. Again, to each our own, you know, pick which, whatever works best for you. The custom attribution API, what is the essence of this API? To explain that, let's actually, let me go to our web app. Instead of going to snowpal .com because it's production data, I'm just going to a local version of it, but you can sign up on snowpal .com, download the mobile apps and check out whatever you're going through for yourself.

So let's say I go here and I go into one of the resources. We have multiple levels of content hierarchy, but let's go to one of the resources that I used for one of the previous videos. There's a number of attributes here. There's block ID, there's name, there's description. This form has simple description, tags, flag. There's a dropdown list of values, associated attributes where you pick a scale, food rating, and then you can pick whatever the other food rate, you know, the scale value is.

Krish (products.snowpal.com) (02:19.214)
So those are related attributes, that's why you see the grouping here. You have due dates, start and end date, kind of a grouping. You have toggles, you have attachments, checklists, comments, list of comments and notes. Comments and notes are similar, yet different because notes are private to you. Comments are shared with anyone else you collaborate with. Attachments is what it says it is. So those are some attributes.

how do, there's two ways to implement this. Someone said Krish, go ahead and implement this as a React or a Flutter UI developer. I'm gonna be like, okay, sure, I need to add, create a form with a bunch of attributes and I need to persist them by calling an API, an endpoint or two or more, but essentially by calling an API in the backend. But if I had to, depend on the API team to actually build this, provide a specification, implement it.

or give me a stubbed out set of endpoints in the meantime, I can still do it, but it's not going to be optimal because again, it takes the API team, depending on how much or how broad and how generic they want for this to be, it's gonna take them that much more time to actually get this working. And it's not necessarily the easiest of things to do. Because you wanna provide customer value by creating and publishing this feature.

What is the point in actually spending time and energy and effort and money? Staffing and hiring a backend team if you don't have one or having the backend team that you might

already have do Something that's already out there and have them reinvent the wheel Shouldn't wouldn't they be better of actually working to your solving your co customer problems? That essentially relate closer to your unique selling propositions. I would think so so

for a number of reasons. And plus, you know, when you have these stub .endpoints, I've seen that when it comes to integration, when the API is actually available, hell, you know, tends to sometimes break loose. So there's a lot more work that needs to happen at the time of integration. And it's expected, your budget and allocate time for it. But all of that is not the best way to spend your time and effort, not certainly in 2024. So with that premise set, hopefully you get the context. So let's take a look at this API and what value it provides.

Krish (products.snowpal.com) (04:35.374)
First thing is different attribute types. So let's say, let's draw, just to take the time to digest some of what you're talking. It helps if we actually.

Krish (products.snowpal.com) (04:50.574)
What are some attributes? There is a text attribute, number, date, file. So let's list it here. These are all the attribute types it actually does support.

that list is bound to increase, but for now, you have text.

Krish (products.snowpal.com) (05:10.926)
I know you have date.

Krish (products.snowpal.com) (05:16.366)
You have number.

Krish (products.snowpal.com) (05:21.742)
File, Single Select, Multi Select

Krish (products.snowpal.com) (05:38.446)
Single select.

Krish (products.snowpal.com) (05:45.038)
And then the last one is nested single select. We will not go into the details of these attributes. I can, you know, we'll do probably cover that in a subsequent video. Nested single select. Because this is a pretty comprehensive API. It's 35 end points, but it's a lot of functionality packed into those 35 end points. Let me maybe change the color of these three.

Krish (products.snowpal.com) (06:14.83)
Okay.

Krish (products.snowpal.com) (06:36.75)
These seven attributes form the... Oops. Oh, because I grouped it. Let me actually ungroup.

Krish (products.snowpal.com) (06:46.318)
These seven attributes form the fundamental building blocks of the custom attribution API. So again, a text is like an attribute such as this way you can enter alphanumeric text. That's what you'd use the text attribute for. You have date. We saw start data, end data, and due date. That'd be date attributes. Number. Well, I think we didn't have a direct straight up number attribute as an example here, but I think maybe if you go pricing point,

Krish (products.snowpal.com) (07:19.918)

Reading from one to five. I mean, that's a number that you enter. So that's a number attribute, even though it's a grouped attribute. And then you have file. We saw file attachments here that we can upload attachments and files. And then we have single select. These are really powerful attributes. Single select is, you know, like the one we just saw. You have a block tab, you can select one of them. That's a single select. Next is multi select is, I don't know if you have multi select here, but if you actually, well, tags are,

of multi select if you add tags. It depends on how you present it but you can treat them as multi select attributes. So I'm gonna say hello and I'm gonna say well it's not multi select from a drop down so it's different but you can imagine this drop down allowing you to select more than one attribute that would be the truest form of multi select. Nested single select is the most complex of attributes. We again absorb all the complexities so implementation becomes super simple for you.

Let me make sure I'm recording, okay. What is a simple example of a multi nested single select? Let's say you're creating a location attribute, right? Let's say we create a location attribute. A location typically tends to have.

a few, it has a country, right? So let's say, I'm gonna say a country could be USA, state, I live in state of Virginia, and then city, right? The city I live in. So that's a location attribute. And again, you can imagine that there could be country, state, and there could also be county, and then city or municipality, district, depending on the part of the world you live in.

That's a multi -select because depending on the country, you want to show a list of states and list of counties in that state and list of cities in that county and so on and so forth. Imagine implementing a nested single -select attribute, a generic one at that. Even if it's not a generic one, your API team will have to do some work. They can kind of hard code the structure to a large extent if you're only doing it for one as location.

Krish (products.snowpal.com) (09:24.622)
But if you have more multi -selects, then that's gonna, you know, you're gonna have to follow a mimic that hopefully you have a design pattern, you still have to mimic it and implement it, do it one more time, and then the third time and so on. If you had an attribute, an endpoint is readily available like this one, imagine how easy your life could be. Here we have fast food restaurants, I think as an example, it's Meg D, Wendy's, and then I guess we have,

I'm trying to see what our example we have actually the restaurants and then we have a state cities and city underneath it. This is actually nested single select. This is the most complex one of them. Before I get to it, I'm just talking about, okay, let me step back.

The nested single select is a single endpoint, but it does a number of things. This representation here gives you a combination of many different things you could potentially do. Again, I don't want to make it the most complex in this first recording. Your attributes or your forms can have pretty complex attributes. The idea behind us supporting these endpoints is to support the broad spectrum of your requirements. But for the one we're looking at, this is a nested attribute, but...

Let me say this, countries is a list of countries, there's only a finite number of countries. Each country, if you take the US, we have a finite number of states, it's 50. And each state has a finite number of counties and a county has a finite number of cities. Now you might have country, state, and county as something coming from another API. Let's say you're consuming a get API or get endpoint, somebody returns a list of countries and states.

Perhaps there isn't one for city, so you have a text field where people can enter the city. So you have three levels, you have four levels actually. The first three are single selects, but the fourth one is a free text field that you type. Or you could have a fourth field that is a number in some other case, or you could have multi -selects along the way, like your final attribute could be a multi -select as well, because you're letting somebody select, sorry, you're letting somebody select more than one city in that county.

Krish (products.snowpal.com) (11:37.038)
We have support for all of them in this custom attribution API. So you have to first create that custom attribute, give it a name, give it your options, customize it. So you're defining this literally a custom attribute that fits into the requirements of your form, the page or the screen that you're creating.

That's the first thing you do. Once you do that, you can then, you know, we have fetch attributes by ID, by name, and updating an attribute, et cetera. Now you can, there are certain rules, you can update an attribute so long as it's not already being used by a resource, or you could have a rule that says it's all right if somebody's using the, if that attribute becomes part of a resource, and somebody created a resource and assigned a value to it. I might still let somebody modify, but it may not make sense. So we have some checks and balances there, because imagine,

adding somebody associated a rating scale and then you got rid of that scale altogether, what's gonna happen to the existing resource that's an instance that's using that particular attribute. So there are rules, so there's data integrity and data validity. But the idea is to make it so simple that you can literally create any forms in really, really quick time, whether you're building a mobile app or a web app or a microservice.

by relying on the endpoints provided provision made available by this API. Let's look at something else, attribute bags. Now you can create a bag of attributes. Now let's imagine that you created an attribute, let's say block ID, name, and description. Let's write it down here.

Krish (products.snowpal.com) (13:20.59)
say actually yeah you know what let's add

Krish (products.snowpal.com) (13:31.246)
We're going to use this for the example.

Krish (products.snowpal.com) (13:40.43)
I don't wanna move this.

Krish (products.snowpal.com) (13:48.91)
doesn't let me move it that's weird okay

Krish (products.snowpal.com) (14:01.006)
The first thing you do, let's call it integration steps.

Krish (products.snowpal.com) (14:10.338)
You create custom attributes.

Krish (products.snowpal.com) (14:16.27)
as required by the form page or screen you're working on implementing. You pick the appropriate attributes. Now if you want, what is a bag, attribute bag?

Krish (products.snowpal.com) (14:31.544)

it's as it says it's a bag of attributes.

So let's in this example here we have name description and simple description. Let's take that name description and then simple description. You can create a list of attributes and then you can choose not to use a bag and then you can go associate the bag associate. Hang on.

associate attribute bags to resource add or update attribute values by bag ID. Let me see. I'm looking for one because you know, we also you can, I'm trying to think. We also have support so you don't actually have to have a bag to, if memory serves me right, to associate the attributes with the resource.

I have to, I'm trying to recollect here because what if I just wanted to not even have a bag? I have to go back and look at this pack again. We have lots of APIs and thousands of endpoints and we implemented this a few months back. So I'm trying to think whether there is an endpoint that lets you actually associate an attribute directly to resource. There may be one, but there isn't one. You have a way to associate a back to resource. You will create a bag with just a single attribute. In this case, we happen to have like three different attributes.

And let's say we create a bag called bag of attributes. We create a create bag. Let's call it, I don't know.

Krish (products.snowpal.com) (16:13.39)
going to give it a simple name and description as the name of that bag.

The reason you create the bag is what if I want to associate the same group of attributes to multiple resources? I shouldn't have to hand pick every single, you know, each one of them every single time. That doesn't, it's not the most efficient way of doing this. So the idea behind providing attribute, supporting attribute bags is you create a bag of attributes. Now I can have a complete bag of these attributes because we have multiple resource types that actually have a lot of these attributes in common. There are differences as well, but there are a lot of them in common. So imagine having to add them every single time. That's

going to be painful, it's not the most optimal. So you create a bag with all of these attributes and then you can drop that bag associated, you know, the bag with blocks, with pods, which are our content, you know, we have three levels of content in our hierarchy.

on the Snowpal web app that is. Using an API, you can create infinite levels of content. There's no, it doesn't have to ever end. So the first level is called key, the second is block and third is pod. In our case, the key blocks and pods have a lot of these attributes in common. So I can create a bag with those attributes and then associate.

Krish (products.snowpal.com) (17:28.686)
bags to resources. That's the idea. Now you've actually associated, created a bunch of attributes, created bags, you can have multiple bags with a combination of these attributes and then you associate them to blogs. Every time you go to create a blog, all of these attributes become part of that resource. That is literally the idea. Now,

Remember I said we've packed a complex set of functionality into 35 endpoints. It took a lot of thinking and the idea behind that is to code Einstein. I think one of Albert Einstein's code is it takes a lot of effort to make things simple. I'm paraphrasing the code, but that's the idea, if I'm not wrong. So we've tried to absorb the complexity so it's easy for you. You can just go implement any levels of these attributes. And again, in the subsequent videos, we'll dig deeper into the nested attribute here, the nested single select.

get as complex as you want for it to be. Imagine having relationships between attributes and making them part of a group and have some rules associated with that group. You don't have to do most of any of it because we provide them to you out of the box. All that you have to do is create a scale for instance, give it a name, create scale values. Each of these scales you can see there's different values. You know, satisfaction levels are love it, like it and not for me. At the same time, if I go to like, I don't know, A to F grades, there's a different list of values.

This is nested attributes. There's two levels of nesting. Again, our nesting supports N levels of nesting, so it can be as complex as you need for it to be. We took one that was up to four levels. I can think of others that can be a little bit more than them, but the majority of your nesting, and it's like the 80 -20 rule, I reckon it's somewhere between three, maybe even four levels, but mostly three. There'll be some that are a lot more and others fewer. You can actually create those, define associate values with them, and then your attribute is good to go. Add it to a bag, associate the bag

resource and your UI team can start consuming it. Now all of this is readily available. All that you have to do is go to like aws .snowpal .com, go look for the custom attribution API. After a couple of clicks you can sign up, get the API key and the product code in an email within the first five minutes and then you start using them to making these requests. It's as simple as that. Now we've also asked have clients who like the functionality but they want it to be provisioned and

Krish (products.snowpal.com) (19:51.984)
that are in that infrastructure, be it Google Cloud Azure or AWS, that's entirely doable, it's something we support. So we'll give provision to our services in your infrastructure by creating a private offer. So you have complete control over not just your data, but the whole request as well. Even some other clients have said, hey, you know what, we actually want to make this part of our infrastructure, but not even, you know, whether we have it in the data center or the cloud, none of all of that,

should be transparent to you. So we want to be able to license this API. So we support that too. So you can license the API, purchase the upgrade license. So we get the upgrades and bug fixes and enhancements and features. And so we wouldn't even know where it is being, which, you know, how you're running it, how you're using it. You're not paying by request or subscription. You basically purchase a license, make it part of your deployment, part of your DevOps or platform engineering teams, and you go from there. That's left to you. So we support multiple licensing models. That's...

just of it.

The way we recommend you get started, again, it's a lot of it is self -serve, but some customers do need a little bit of assistance at least to get going. So we have professional services for each of these APIs. You can purchase them on AWS Marketplace. So it becomes part of your single consolidated billing. Or if you want us to put together a private offer, we are happy to do that as well. So this is again, the custom attribution API. There's articles that we have on products .snowpal .com. If you can go to API, you can filter and look through

We have a of articles related to APIs and some speak to the custom attribution API. Now that's sort of a high level overview. And if you had the API and was not having it, you can imagine your UI team wouldn't have to wait on drops from the API team. You don't need stub .endpoints. Stub .endpoints is okay, but it's painful because you're going to call it, you can start building it. But when you go to integration, you're going to find a lot more gaps and issues that you have to address. And that's

Krish (products.snowpal.com) (21:54.064)
That's a risk that you can avoid. If you can't avoid it, why not avoid it? And that is literally the idea behind us publishing these APIs as fundamental building blocks. There's a lot more we could talk about the specifics of these attributes, but I actually think we might be better served.

if we sort of defer that to the subsequent recordings but take a look at these endpoints you know if you have questions let us know look at them in the context of our developer documentation we also have SDKs published for our APIs we have Golang SDKs but that list is you know we are beginning to support other

languages that's in our roadmap for this year, TypeScript or whatnot. So you can go to custom attribution SDK. It'll take you to package go .dev and you can look through our examples and recipes and whatnot. So this SDK, we have an SDK in Go, like I said, for all of our APIs.

Okay, I think that's a fair point to end this first video. Hopefully that answers some questions for you about our custom attribution API. If you have more, definitely hit us up. Talk to you soon, bye bye.