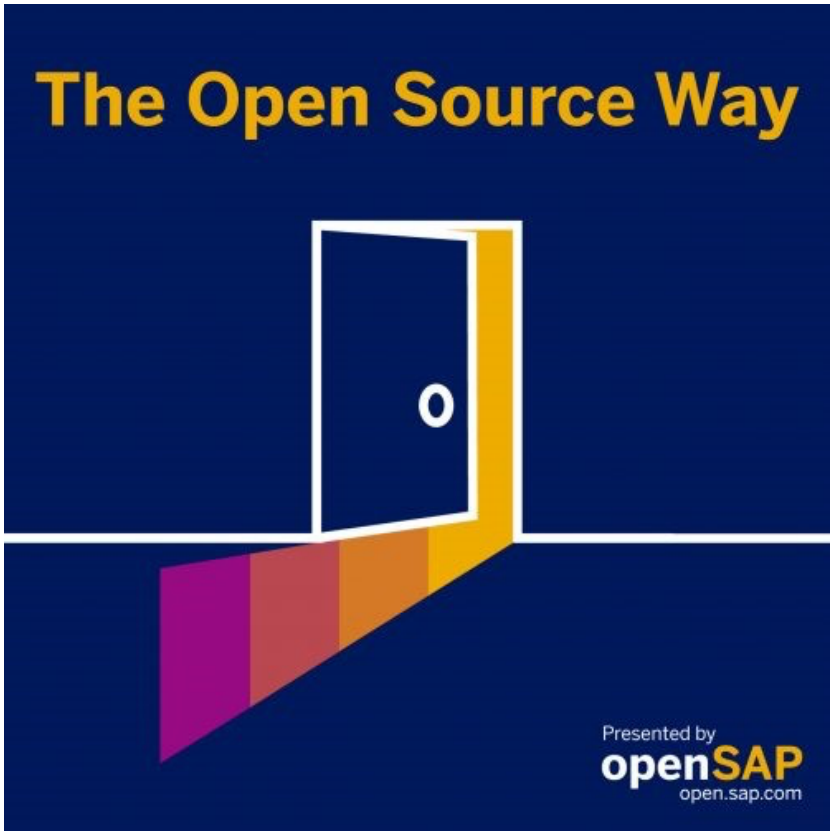
The Open Source Way

Episode 12 – SAP Cloud SDK



Transcript

Karsten: Welcome to the Open Source Way, this is our podcast series, SAP's podcast series about the difference that Open Source can be. In each episode, we will talk with experts about Open Source and about why they do it the Open Source way. I'm your host, Karsten Hohage, and in this episode I'm going to talk to Marika Marszalkowski and Frank Essenberger about the Cloud SDK. Frank works as a senior developer in the SAP Cloud SDK team in Potsdam. And like many other colleagues at SAP he has a science background as a physicist. He says that besides SAP and his family, he doesn't really do much because that takes up all of his time - with his two children keep him rather occupied. Marika is a senior developer in the SAP Cloud SDK team in Potsdam, as well. She studied IT systems engineering at the Hasso Plattner Institute, and her most unusual hobby is ice skating. But beyond that, there is something that they also sometimes, or actually every day, for all I know, do together... Marika, can you tell us about your special team events?

Marika: Yeah, we play half an hour of "Doppelkopf" every day, And it's actually a lot of fun. It's a game that usually only old people play in bars in Germany, but it keeps us together and also through those times with Corona.

Karsten: Did you just say old people play that in bars?

Marika: Usually, from what I saw.

Karsten: Okay,

Marika: What I saw.

Karsten: Okay, maybe the classic card games have a tendency of being forgotten, that might be right. Do you know if there is even an English translation for that? Doppelkopf? "Double head?"

Marika: I don't know.

Karsten: Uh,

Frank: I don't know.

Karsten: Or "Sheep's Head" because it's called "Schafskopf", as well, isn't it?

Marika: I think it's a similar game.

Frank: Yeah, I think "Schafskopf" is almost the same, but with a slightly southern German touch. So, I think in Bavaria and Baden-Württemberg they call it "Schafskopf", but it's not exactly the same.

Karsten: Okay, I wouldn't know, I've always been a "Skat" player; of which I also don't know if there is a translation for that. But we're not here to talk about card games, right? We're here to talk about the Cloud SDK. So let's maybe start with a short description. If you could do that in two or three sentences, Frank, maybe, what is the Cloud SDK?

Frank: Yes, sure, so I'll try to keep it short. It's a library, an NPM library which helps you to develop web applications on SAP's Business Technology Platform.

Karsten: Okay I saw that there is a Java and a JavaScript SDK, is that the same approach or is it actually the same thing? Or how do we separate?

Frank: Yeah, so the Java version was there first. This all evolved from a product called RealSpend, where we realized building blocks which were most likely relevant for all developers doing web applications. And this was written in Java. So the Java SDK was first, and then we adapted it for JavaScript TypeScript world, and we aim to have feature parity between the two. But in general, the Java team is sometimes a little ahead because they simply start before. But I think we're now in a spot where we almost have feature parity, I would say. Right, Marika.

Marika: Mostly, yes, there are some parts that we will probably never do, they support the older SAP Neo environment on the SAP's Business Technology Platform. So that will never happen for the SDK for JavaScript.

Karsten: Now SDK for Java kind of sounds familiar, not only from last year, so I just assume there has been a history to that. Is there like a line of predecessors to the current SDK approach?

Marika: Yes, you could say so. So, as Frank already mentioned, the SDK or the SAP Cloud SDK for Java, first of all, started as a carved out part from a product that we developed on the, back then, SAP Cloud platform. So when we started working on that, this was one of the first Cloud applications that we developed. And this product was called SAP RealSpend. And we realized that what we were doing was probably something that many people wanted to do, which was building a Cloud application on the SAP Business Technology Platform today and connect to the SAP S/4HANA systems. And we saw that there was quite a lot of boilerplate code that might be necessary for this. So this was carved out, and back then the name of the SDK was actually a SAP S/4HANA Cloud SDK, because the idea was to enable this connectivity specifically, so for extensions. Now, it is a little different, the scope is a bit broader. We do not only support S/4HANA, but also other systems, and therefore we changed the name.

Karsten: Okay, Now, Frank, can you briefly explain what NPM libraries are?

Frank: Sure, so NPM stands for node package manager. So this is the dependency, your package manager, when you run Node, and Node is the standard engine when you execute JavaScript outside of your browser. So when you generally extend your back end with some web application, right, which also runs on JavaScript, you execute this with Node and then the tool to manage your dependencies is NPM. It's similar to Maven in the Java world. So the standard server where you can get all your dependencies from. And there we publish our SDK.

Karsten: Okay, so the SDK manifests itself as NPMs?

Frank: NPM is just the term for the package manager, right? And when you just include in your project, you want to include any dependency. I don't know what it could be; what's a standard NPM dependency?

Marika: HttpClient.

Frank: Yeah, HttpClients, right. So somebody has developed a nice library, so you must get it into your project in a convenient way, so you don't want to go to the home page and download some binaries or so and put them into your project, but you want to have a server, some repository, where you can get - with the version specifier - the version of the HttpClient or the latest version of the Cloud SDK, so you can have nice S/4 connectivity, and you then say, okay, NPM, here's this repository, please give it to me. I think this is mainly hosted by Microsoft, who pays for all the hosting, so everybody, every Open Source project and so on can publish libraries there, and if the people like them, they can download them and use them in their project depending on the license. Yeah, also like in Unix, when you have your distribution, you have different package managers like APT or some other package managers as a central repository in the Internet where the libraries lie and you can just access them from there.

Karsten: Okay, I guess everyone out there who's listening knows anyway, and I'm the only non-developer in this podcast on the talking and audience side anyway. So, let's maybe instead go for a deeper look into the SDK. Marika, can you explain in more detail what the Cloud SDK does?

Marika: Yeah, so first of all, the Cloud SDK consists of a lot of NPM libraries, so the packages that Frank just described. And currently, if you look at the total of the libraries, there's around three hundred, I think. So it's quite a lot of libraries, although the most important ones are a handful, I think it's around five. So basically, you can separate the SDK into three major parts. And the most interesting for today is probably the one that is Open Source, because not everything of the SDK is Open Source and the major part that is Open Source is basically what we call the core and the generators. The core is responsible for connectivity on the SAP Business Technology Platform, so for allowing connectivity through the destination and access UAA service, for example, and also allowing or giving a basis for clients that we allow to generate with the generators

Marika: The generators are mostly meant to support... to support usage of APIs. So SAP provides APIs to access different kinds of systems. As I mentioned before, historically, we were focused on S/4HANA, but there are other systems and other services to look at. And our first generator, the one that we have had for a while now, allows you as a user to generate your own client to access those APIs in a typed manner, to access them from code, which is a bit easier than building up URLs on your own. So, basically, it's a sophisticated URL builder for specific services that you create. The first one that we had, historically for a long time now, is meant for OData, for both existing versions v2 and v4. And the second one is for OpenAPI, which is the open standard for APIs, which also covers a broader scope than than OData, I think, also outside of SAP. And that is the big, most important part, the part that is Open Source and is available under the Apache 2.0 license. Then, there are two other quite big parts, but those are a bit different or actually one big part and one smaller part, I would say. The second part is pre-generated libraries, so we basically use our own generators to pre generate libraries for the SAP services that I just mentioned, and those are closed source. So they're free, you can use them without charge, but those are SAP IP and therefore have to be under a different license, which is the SAP developer license. And those are actually the major parts. There's a third part which is very new to the SDK for JavaScript - Java has had that for a longer time already - those are extensions. So currently we have one extension, which is meant to provide libraries for currency conversion. And this is also Open Source.

Karsten: Okay, so in summary, for stupid me, SAP has an API, and then with a generator, you either have that pre-generated for use in the SDK or users of the SDK can generate themselves the access to that API into the SDK.

Marika: Exactly, and you can generate this not only for SAP APIs, because it might make sense to just use the pre-generated libraries that are available through the NPM registry that Frank mentioned, or generate your own for your own services or services that we do not provide, potentially.

Karsten: Okay, I think I got it.

Frank: I would like to comment that for the S/4 system, for example, we have generated all available OData services, a few hundreds, and you can just download them. But if

you, for example, have your custom, own service, right? Of course, for this one, we cannot offer anything, you can generate, or if they added a lot of fields and additional content to an existing service, this is also possible. If you don't have a Cloud but an on-premise S/4 system it also makes sense to regenerate. And then you have all the fields nicely typed and all the new what you added there in a nice type-safe way. Or if you have any other system which is OData or REST OpenAPI you can use the generator to get a nice client.

Karsten: Okay, and as we're talking about this being an Open Source project, would generators be something that people who use it then provide back or contribute, I mean?

Frank: I mean, we had a library which goes a bit into the direction; it did not use the generator, it was this currency conversion use case where another team used parts of the SDK and it became part of the SDK. The problem there is and that's also the reason why we don't ship generated client under this Open Source library that, for example, the business partner API is the intellectual property of SAP, so the summation of first name, last name, date of birth and all that. That's why we have to split. So all the tools for generation and usage are Open Source. But the specific data model is published under the developer license. You also find it in NPM, but with a different license. So if some other SAP colleague would like to publish it, they would either have to change their license in a way that it's also Open Source, this library, this API definition or we cannot put it under the same project because I think you have a license per project and not a snippet of code, unfortunately.

Karsten: I see that there are some issues to watch when licenses are concerned; what was SAP's motivation for running the SDK as an Open Source project in the first place?

Frank: this came mainly from us, from the developers, and I would see perhaps three main things which would put us into the direction. So one of the main things was that we wanted a more direct contact with the customer. So when you have Open Source, you are in the Internet, everybody can see your code base. And if a user or a customer has an issue with our SDK, cannot connect or whatever, they can directly make up an issue, ping us, can have a look at the code, can understand what's perhaps happening. And this is not the case when you are not Open Source because then you have different

support channels which are much more indirect. And this is also true for the release process. So when you're Open Source, right, you're on GitHub. And everybody can just see what we make every week or every day, it depends on what we what we have done, and see when we release new features. You just see that there's a new version, what are release notes, what are the new features

Frank: . And the last thing I would say, I mean, this is also something I learnt from this very prominent Open Source project, Corona-Warn-App, right, that's also from SAP and that is also Open Source. This public exposure is very good when it comes to your code quality. Because as a developer, you are constantly exposed, so every PR, every comment I write could be read by other people. I comment on something from a colleague or I see a potential issue and also other people could see it. If you are a handyman and your work is publicly displayed in a showroom, you want to present your best work,because it's very transparent.

Karsten: Okay, so I see we're kind of getting into the general Open Source argument here, right, that actual quality control and support are better in the community than having an x to y defined relationship, right? Speaking of our "classic" customers, though: Have they adopted that perspective in your experience? Because it seems that we sometimes still have customers out there who prefer the classic support channels for things where they have a guaranteed answering time and so on, and they rely more on that than on a community.

Marika: I would say that, at least for us, I can say that most of the issues come through GitHub issues, so directly through the Open Source channels and most of the issues are also interactive. So people are asking back and forth and are interested in the responses. Of course, sometimes people just dump something and then it gets lost. Maybe it wasn't that important or something like that. Usually, if you use the classic support channels, you are also definitely interested in the response. So I have never seen that somebody forgot the tickets there. So that doesn't happen that much. But I would say the load is higher for us on GitHub issues. In comparison to Java, our Java brother, basically, we are the only ones that are Open Source, so Java is actually also planning to maybe follow us sometime in the future. We don't know that yet. But they're at least looking at this and they're actually interested in that. And they, of course, get much more support for that, although they support other channels like Stack Overflow.

But I think their main support channel is still the classic SAP support channel, and being Open Source also leads people to acting Open Source when asking for support.

Karsten: Yeah, so it seems that in the developer community, basically, it doesn't really matter if you're on the side of the customer or the provider or the consumer or the provider, really. But everyone, as you just said, lives it Open Source, right?

Marika: Yes, we were actually worried about that, that it might be an issue And that's also why we definitely support the classic support channels. But we also prefer the Open Source channels just because they're more interactive and more clear and direct. And you can actually point out specific lines of code where people think there might be an issue with something, so that's very helpful.

Karsten: Well, then that seems to work. Let me ask the other way around: Were there any specific issues along the way when you went Open Source with that or what were the biggest pitfalls that you can basically point others to so that they can avoid them if they are thinking about open sourcing something.

Frank: I think what you have to do before is to look into your code and inspect if you have some legal issues. What I mentioned before was these parts which are intellectual property, you have to be careful. I mean, there's a process within SAP that you start with, let's say staging repository, which is on GitHub, but it's not publicly visible and this can converge. And at a certain time when you think, okay, now everything is safe, I don't have any references to internal systems or any backlog references to internal communication or some secrets which you're definitely not supposed to show to the world. When you converge there, you can say, "yes, we are ready" and then this staging repository becomes public. And what I think we definitely underestimated a bit was how much information we use in our daily planning and our daily backlog that we cannot expose. And this is also a thing, I mean, where I would like to have more transparency, that the users could also see what's our backlog, what we are currently working on. But there's so much internal information in our backlog that we cannot make it public, perhaps this will change when more and more parts of SAP's cloud product's eco-system become Open Source,

Karsten: Yeah, I guess, I mean, it starts with the issue that as soon as you publish your backlog, somebody might take this and consider it, as promised delivery, right?

Frank: Yeah, I mean, there you could at least write a big disclaimer, but I mean, very often, we have internal things with our release, with our SAP internal infrastructure, discussions, mentioning names of other colleagues. Referencing back and forth, emails and so on. GitHub has to do is offer some support so that you can always blacken stuff, mark it as confidential so that it's not visible.

Frank: We started with a closed repository and of course there are then things which you just dumped, And you have to find all these points before you go public.

Karsten: Marika will probably have to add a lot of detail here.

Marika: Maybe not a lot of detail. Frank already painted a good picture. we, of course, have tools that SAP requires for products to be there. There are standards, that's correct, that SAP has to fulfill. And these products or product standards also apply to Open Source products, of course. And internally we have certified tools and also tools that SAP pays for, expensive services that are running and doing stuff to check your code. this, of course, is not available in the Open Source world. So it's not even reachable from the outside Internet. And what we had to do was find replacements for this, and also check whether those replacements are valid replacements. Sometimes, in my opinion, some of the replacements are even better, but others are maybe a bit worse. So that's a bit difficult to balance. And I think this is something that SAP in general still needs to work on, which SAP is actually doing. I just got an email a few days ago that we are now part of the program to check GitHub security tools that are there to check whether we can certify them even internally. So externally would be, of course, great in the Open Source repository, but internally, for the internal libraries, that would also be very helpful.

Karsten: Let's maybe turn to what are the biggest improvements that have come about with being Open Source?

Frank: I mean, we've already mentioned the support and the direct contact to customers and so on, but one thing I didn't think about was when we were hiring new

colleagues. So, when we had interviews with candidates, it happened that the candidate could just look, what is the code or the project that the potential new colleagues are doing. So when they were interested, they saw "ah, okay they're currently implementing some new X, Y, Z feature, were doing the reviews like that, this is their stack, the technologies they're using." So this was very cool and also then showed that the candidate was really interested. And then also, once they join, it's a very standard tooling, so GitHub and the tools, if you have done already some projects, even if it was your TGI Friday Project, you know the tools, it's not something closed that only SAP uses. So this also speeds up the onboarding of the new colleagues.

Frank: In the beginning were had a few problems using some libraries, which turned out to be solvable. This was also very positive. And then you check all the libraries you use and see if they are all suitable to use in Open Source. And if not, you have to find replacements. And it turns out that the Open Source world is very big. And we find even better libraries to replace the ones which are under license, that you could not use with our Apache license, with. And so this was also very pleasant surprise. And in most cases, when we had an issue - I had some security stuff and so on - the time to fix these in the libraries we used was very quick. So you could also say if you don't pay for it, how long does it take for them to fix an issue in some public Open Source, but it was always very quick. I mean, there are exceptions, but on average.

Karsten: How about documentation? Does that work well?

Frank: I think it's rather standard as GitHub pages, right? So when you have a GitHub repository, there's sort of related to it some GitHub tool which creates some documentation for you. And we just use it. And I think it looks rather decent without much work, there's a good search on it when you search for keywords and so on.

Karsten: Okay, that's kind of a standardized way, it seems. How about the learnings when, for instance, you have to replace a library due to license reasons or something? Do you document that also somewhere or is that just inherent in the code then?

Frank: I would say it depends on the size of the change. If it's just a small replacement, it's in the code. If it's a bigger issue then there's of course some architecture decision records, saying we want to do that because and some discussion. And then there are

bigger documentations on these changes, because very often if you replace a bigger library, this most likely also comes with some behavioral changes. So you have to see if you can keep your API contract or you have to introduce a breaking change. So a new major version; there you have to be careful, right? I mean, if you have a few thousand downloads a week, you already realize when you change something in your API - even if it's not a feature which you publicly communicated - some, let's say, hidden thing, people rely on it.

Karsten: So they find that.

Frank: Yes, yes, I mean, once it's out, I mean, even if it's not part of your public API, it's always sorted. It's not specified that it's sorted, but it has been. And if you change it, they will open it issue saying that it's not sorted anymore. And so you have to be careful.

Karsten: I mean, we're even speaking Open Source here, and I know they even find the APIs in the proprietary software that were not documented and not meant to be ever used by anyone. Let's not speak about that one either. We've spoken about the documentation that leads me to the question: Where do people go to get started when they want to use the SDK, get involved in the SDK? Just the GitHub pages, or is there anything else you want to point them to?

Marika: I think GitHub pages is a good spot. Do I know the URL from from the top of my head, it's SAP.GitHub.io, search Cloud SDK.

Karsten: And I'm pretty sure we'll have that posted under the podcast once we publish it. Right! Okay! I think we're almost at the end of the things we wanted to talk about. Maybe you have, for everyone out there, your three main things or takeaways that you want everyone to remember from this podcast.

Frank: Well, ladies first, please.

Marika: Sure. Okay, I think the first thing that's probably quite generic, but nevertheless something that I want to mention, because I think this is very important and one of the major takeaways that we had at least, are transparent support channels. So through this transparency that we now got for ourselves, but also for the users, we overcame a few

issues. First of all, the many channels that we had before. Before, we supported different channels. We still support all those channels, but we still focus on the issues in GitHub, so that even if there was an issue somewhere else, we create our own to track it. So there's basically one source of truth, and the other problem that is solved by that is trust. So I think the transparency that we have through the support channels currently gives customers bigger trust

Karsten: That's one.

Frank: For me, it would be this, I would call it self discipline, so that you are exposed and you always try to be the best you can be; be polite, deliver the best quality, and always try to improve yourself.

Karsten: That is almost like a wisdom with a picture to be posted on the Internet. That's two.

Marika: Okay. And the third one is that we started off on a bumpy road with this whole project; starting internal and then moving it to Open Source. Some things were harder, some things were easier. But I think - and we experienced this also with the extension project where we supported - that, once we paved the way, it will get easier. So every new project will have it a little easier.

Karsten: thank you very much, Marika, and thank you very much, Frank, for being our guests today. Everyone out there, thank you for listening and I hope you'll be back. There's a new episode out every last Wednesday of the month. And you can find the episodes and the whole list of episodes on all the usual podcast channels on openSAP, but also on Spotify, Apple podcasts, etc. Thank you again. And let's all say goodbye. Bye bye.

Frank: Yeah, thanks for having us, bye.