Krish (products.snowpal.com) (00:00.906)
Hey there, hope you're doing well. In this video, we're gonna take a look, reasonably quick look at one of our APIs, the Access Control List API. And we're gonna talk about some of its features and I'm gonna tell you how you could possibly use it, integrate that API and actually save a lot of time, effort and money and not reinvent the wheel from an access control dependency standpoint. You can have your web mobile or service site teams work on your core customer problems and simply integrate the API and not have,

you know, do the heavy lifting of implementing it. So we're just gonna take a look at some of its features. It does a number of things. We'll get to as much detail as we can in the next 10 to 15 minutes. And then if you have more questions, you're welcome to hit us up and we can go from there. So let me actually share my screen.

Krish (products.snowpal.com) (00:52.174)
Okay, let's take a look at our existing app. Our production app, one of our products is a web application, which is, you could use that for managing projects at school, work, home, and everywhere else essentially. You can go to snowpal.com to check it out. Given that it's production data, I'm just gonna go to a local deployment. So just ignore the quality of this data that's just running on my machine.

But to give context to the API that we're looking at, I'm not gonna go into the details, but when you create content on our project management app, you can do just about anything you can expect to be able to do on project management solutions and beyond. One of our pieces of content is called blogs. It's the second level in our content hierarchy. I can actually share this with anybody. In this case, I've shared it with a couple of users, Parag two and three.

Again, it's dev data, that's why the naming is as dev as you can expect for it to be. Parag 2 has write access, Parag 3 has admin access. And this is me, since I'm the one who's granting access, I happen to, I need to be admin. That's our basic ACL model. Now, the API we're gonna look at is the access control API, and you can go to acce where the first access control list API is like dash delimited.

or you can even check out our developer documentation by going to deve I'm gonna focus on the Postman collection just for the purposes of this video. And remember this, in this simple example, somebody's asking to implement a web app, your React developer or React UI team is gonna go start working on building this implementation or something like this. If they have the APIs available, the endpoints available, they can hit the ground running.

you know, the basic model is defined, you publish a specification, and again, start coding to it. Implementing this page with the APIs being available already is gonna be a whole lot easier and quicker. You know, with React, you have tons of components. All you have to do is piece, you know, put the components together, understand your requirements, call the backend services, and you're good to go. But if you don't have the, let me reduce the volume a little bit. If you don't have the APIs,

Krish (products.snowpal.com) (03:09.634)
the endpoints it's actually going to you know it is going to take a lot more time the idea behind us publishing our API's as build a fundamental building blocks and one of our API's actually called building blocks but the rest of them are building blocks as well they just help you build different aspects of whatever it is that you're building it'll actually help you know hit the ground running really quickly so if you took this as a requirement you know read write an admin as three levels of access control

We have three privileges. So you can actually go in to this endpoint. Let's say all you have to do is the essential mapping, the member privileges. You essentially create, there are three privileges. We support the notion of custom privileges. So you define three privileges, read, write, and admin privileges.

Once you create those three privileges, we don't have a notion of roles, not yet in our existing implementation. Role is essentially a bag of privileges. Actually, let's jot down some of these terminologies here. It'll make it easier. Let me go here.

Krish (products.snowpal.com) (04:25.347)
control API

Krish (products.snowpal.com) (04:33.151)
Got a different shape.

Krish (products.snowpal.com) (04:40.014)
privileges. We saw three of them that's the lowest level. We saw read write and

admin has three privileges. Those are just our, the snowpal.com's basic ACL model. Our API lets you do, create the simplest of access control for your application to the most complex that you can possibly imagine. The next two roles, a role is a bag of privileges. So you can bucket all the privileges into roles. Actually, let me do this. Let me move it here.

roles and privileges. Let's give this a different color. We're going to call it Members.

Krish (products.snowpal.com) (05:26.994)
teams. Hopefully I'm sharing. Yes I am. Okay let's just draw the arrows

Krish (products.snowpal.com) (05:42.574)
Okay, what is a role? Role is nothing but a collection of privileges.

Now in this example we saw, we just saw read, write and admin. Read is essentially in our world, anybody who has view access to this piece of content, to grant them right, then they're able to make edits and changes to the content and admin, once you have admin, every all the access you possibly can have, which is in addition to read and write, you're able to grant access to those resources to other users and grant them access as well. That's our ACL model. So roles is a collection of privileges. We don't have a collection right now

implementation and the web implementation. Our API clearly supports that. So privileges, let's come up with different privileges to understand rules a bit better. Now when you go to this content here, there's a lot of different attributes and one of them is say a due date and let's pretend that we want to have specific privilege that says edit all attributes, edit due date only.

edit description you know just this is manufactured so you can take that with a grain of salt now you have a collection of privileges for a role so you can create a role that's basically let's call it

Krish (products.snowpal.com) (07:08.302)
Roll.

Basic edit role and this role will let somebody edit due date and edit description

Krish (products.snowpal.com) (07:23.818)

It'll say the next one is advanced edit role is the name of the role. It lets you edit all attributes. Let's pretend, say if it's a single privilege and the role may not make sense, I'm gonna have to make this a little bit better. So edit all attributes, edit due date, edit description, edit tags. Okay, advanced edit role could be, let's forget the edit all attributes, edit tags.

and there could be more added comments or something. We have more attributes. So the second role has three privileges. The first one has two. So what you would do here is essentially go create those privileges.

one, two, three, four, five. So create privileges is the first thing you would do. And after that you would create a role. And when you create a role, you can associate privileges with that role while you create the role. Or you can just go associate privileges to the role once you create the role. I'm not going into the details of the actual endpoints. We have thousands of endpoints. I obviously don't remember all of them. You can go look at the specification, which is what it's meant for. I'm going to assume they support a lot of the attributes.

I'm going to socialize with you here but the truth is literally what is stated in the specifications up to date so just take what I'm saying the grain of salt and trust the specification as it's defined and published to you. So associate privileges to a role so first we create a role and then we'll associate the privilege to this role so there are the list of things that we would actually do let's actually try to create a simple maybe not really I don't want to grid I

list.

Krish (products.snowpal.com) (09:16.546)
Yep, let's do that.

Krish (products.snowpal.com) (09:25.806)
worry about the format of it as much. Okay. ACL. Integration steps. This is what your UI teams experience is literally gonna look like. They're gonna review the access control specification.

Krish (products.snowpal.com) (09:45.858)
Review, access, control, API.

specification.

Krish (products.snowpal.com) (09:54.706)
And by review, I don't say they're going to spend hours, they're going to just spend a few minutes here, get a feel for it and it's pretty self descriptive. We also offer professional services. If we need help, we are more than happy to help you, you know, hit the ground running as well. Okay, so you, what are the things we're doing? Review that and then they start, let's say we'll talk implementation, create privileges. In this case, we had five of them, create

roles associate privileges to each of those roles

Krish (products.snowpal.com) (10:35.902)
Now, our implementation right now currently as it stands, as I showed you here, let me go share.

It's very straightforward. We want to keep it simple, intentionally so, to the target market that we are selling our project management application. But the idea behind our API is to be able to build the simplest of access control, to satisfy the simplest of access control requirements to the most complex ones at the same time. So we did not have roles, like I've said more than once, but in our example, we have two roles.

and we associated the privileges. So we keep chugging along. Now what are, let me go here. What are members and teams? So here we associated, we granted, as we saw, we granted Parag two and three access directly. So they are members. So members are actually users who you grant access to. Members are users who, access control is grant.

Krish (products.snowpal.com) (11:48.574)
Okay, what are teams? Teams are much like what we saw earlier with the roles collection of users.

Krish (products.snowpal.com) (11:59.522)
So if you wanted to grant, let's say, let me copy the same note, the same color I meant. You could say, you could create teams called write users and other team called read users.

Krish (products.snowpal.com) (12:21.022)
and you could say parag one and parag two are right users. And then you could say parag three and parag 10, four, let's keep it simple, are read users. Again, this is the manufacturer as you can tell, but you know, hopefully it gives you the idea. Those are two teams that we've created essentially. So what we did here was after that, we went ahead and created,

Krish (products.snowpal.com) (12:52.43)
Here we can go to, let's collapse this.

Krish (products.snowpal.com) (13:00.278)
So the members are essentially existing users. So they are users of your system. Now those users could be coming from a different identity management system altogether, or you could use our APIs to create those users as well. So we have users where you can actually, the creation of the users is part of a different API. So you can either you combine another API and create these users and sign up. Like you could go to the, let's say a building blocks API for instance.

our project management or content management we have, oops.

Krish (products.snowpal.com) (13:40.458)
postman there okay so there's a building blocks API and you can actually go into users and you can sign up you go to registration and you can sign up a new user

Actually, you can do that here as well. Let me take it back. I forget, we actually, all of our API support the notion of registration, so you can sign up for users independently so without combining APIs, or you can use users that come from other systems and only use access control here from managing your authorization side of things. It's entirely up to you. But let's say you create users by registering them, and then after that, you can create teams. If you go here,

Um...

Krish (products.snowpal.com) (14:25.838)
That is a, oh, create teams, create team is here. So you would go call create teams. And then after you create a team, you would basically associate members to a team. That's like the associated privileges to a role. Plus to each of these teams. So, I'm gonna go ahead and create a team.

Now that you've created all of that, you can start assigning, you know, granting resources. Now, if you go down here, we have resources. Now you can grant, let me go to Fetch Teams,

Fetch, Team Rules. You have a lot of these endpoints. Let's actually go to privileges, member privileges, because you know, in terms of hierarchies, you can imagine it's, here is how I would write it down.

Krish (products.snowpal.com) (15:15.632)
it's member privileges simplest thing much like snowpal does assigning privileges the rewrite etc to members or you can assign privileges to teams or you can create roles and assign roles to members or you can create roles and teams and then you can team okay sorry member privileges

Krish (products.snowpal.com) (15:45.038)
team, oh hang on, what am I doing? Team roles, member roles and team roles, right? So again, there's not really a hierarchy in that sense. You know, I'm just saying the simplest thing to do is create privileges, assign it to members. The next simplest thing to do hierarchy, maybe in terms of simplicity, if you will. Yeah, privileges, assign that to teams or it could even be, maybe this is a better order. I don't know.

Yeah, we can create roles and assign to members and then you can have privileges assigned to teams and roles to teams. I'm just saying that there's four variations of it. So it can be as simple as privileges to members or it can be as complex as roles to teams because roles is a collection of privileges and teams is a collection of users. So to me, I see this as a simplest one and this is the most complex in terms of your structure. After you create those assignments, you basically take a particular resource and we have these

What is it? Assign roles to a team, fetch roles, blah blah. Let me actually go to resources.

add or update a resource. You can check fetch nested resources and then add nested resources. You can look at the end points more carefully in our documentation that supports this as well to get a sense of what each of them is specifically going to be doing. But it should suffice to say for this initial video that you can take any resource, resource that you create on the Snowpal using Snowpals APIs like I mentioned earlier. Our APIs can be used independently.

perhaps not in this video in a previous one, can use their APIs independently or collectively so. So if you go to projects, project management API is one of the APIs, you can create content like, you know, actual projects and cards and then assign privileges, you know, permissioning associated with those resources. Now we can use our project management API to actually create the resources and bring those resources and grant access on those resources using this API, or if your resource creation happens natively

Krish (products.snowpal.com) (17:50.404)
APIs or you're integrating other third-party solutions that's alright you can just get the resources and use the resource ID to essentially grant access on those resources through our access control API that's the idea you bring the resource ID here I believe what do we have here our resource I mean we have a nested resource it's location

location could be USA as a country and then state of Virginia, if it's in Virginia you have Reston and Herndon. So this is a nested resource and you're granting access, some kind of access, either privileges or roles to teams or members to this resource. You have a resource and then you're granting access on that resource, you're granting certain level of access. That's what you're doing using this particular API. Once you grant access to this,

Let me close this to this resource. Now you can then fetch, make the request to do the fetch. So let me go, let's say team privileges.

It says assign, what does it say? Assign privileges on resource to a team. If you go to like member privileges, it's gonna say assign privileges on a resource to a member. So you basically pass a resource ID, a name and a title, and then you pass a certain level. We log into the details of what level is, not in this initial one, it's just gonna make it a little bit more complex than it needs to be. It just gives you more power in actually structuring the hierarchy of your access.

that's what a level is. So once you grant access to it, you can then say, you know what, fetch privileges assigned to team on resource. You can say fetch highest privilege levels on resource for a team, which means, you wanna know whether somebody has admin access, it's a custom privilege or a custom role that you create. But in the case of the Snowpal implementation, if I wanted to know, hey, who has the highest level of access, the highest access is defined by admin, who has the highest access

Krish (products.snowpal.com) (19:58.184)
this particular fetch highest privilege level on resource for this team. So you could have many teams that have access to a particular resource. Each of the team has any number of members associated with them. This endpoint is basically saying, hey, get me the highest privilege level on this resource for a given team. For this team, product team versus the

Krish (products.snowpal.com) (20:28.416)
And they have a lot of privileges, but different levels, privilege level IDs, get me the highest privilege because I'm gonna make some determination based on that response. If they have that privilege, I let them do something. If they don't have it, I don't let them do something else.

That's one example, but you can imagine how complex this implementation can possibly be. But the idea behind us publishing these APIs is so we can deal with all of the complexity ourselves and have you and your UI teams actually built to integrate these APIs and not worry about implementing this. You know, just take a moment to think about defining the specification, implementing it, making, deploying it, upgrading it, enhancing it, making it scalable, and doing more and more and more.

that is not gonna necessarily add value to you because that's probably not your core competence. You're trying to solve a different business problem. You should focus your energies in solving that problem and leaving the rest of the backend heavy lifting that's pretty agnostic and fundamental and generic to us. So in terms of integration steps, we have this, let me put that in, okay.

So you created the teams and then let me go back here. Now we're gonna associate, I'm gonna say either create a resource if you have to, then you're gonna associate, I'm gonna say grant access to resource. Okay, grant access on resource to members are.

teams and where access can be defined as privileges or roles.

Krish (products.snowpal.com) (22:07.37)
and it would be a combination. A single resource can have N number of privileges and roles granted to members and teams. So the actual structure can be as complex as you need for it. Imagine this where you're actually granting privileges to members, privileges to teams, roles to members and roles to teams. So this page can get as complex as needs to be, but your UI team only needs to worry about designing and implementing and making it friendly for your end users, whether it's a React page or a Flutter page, whether it's a mobile app,

it does not matter or you might be building some other microservices that have access control dependencies you just delegate those dependencies to our APIs and not worry about reinventing the wheel. So that's so you grant access and then you check at runtime and you know when somebody's going to access something check if user or a check of member has relevant and necessary access

to resource. I'm actually to realize, hopefully I can minimize this, but I know this, let's mean, okay.

Krish (products.snowpal.com) (23:17.602)
I want to reduce the size of this but I don't know how this tool supports it.

Anyway, so you can check if the member has access. Now imagine having all of this readily available to you. Super simple, right? And that's how it should be. All that you have to do is either go to aws.snowpal.com and it'll redirect you to the Amazon's marketplace. You can go to access control API, do a couple of clicks and you'll get the API key and the product code and you can start using it. You can pay by request or subscription. It's super affordable. And if you want us to put together a private offer, we are happy to do that as well.

clients have asked us that, hey, we like what you offer, but we won't be able to manage that in our own infrastructure. We are able to make that available, be able to provision this in your infrastructure that's on AWS or Google Cloud Azure. Or you can simply license the API and have us be out of the picture. You purchase the license and then the upgrade license as you deem appropriate. And you can integrate that into your code base, repos, and your whole DevOps infrastructure.

Krish (products.snowpal.com) (24:24.77)
It could be as simple as you leveraging APIs as a SaaS model or essentially you purchasing licenses and making it part of your infrastructure. There's many ways we can make this happen. The idea is to give you all the time back, mitigate your risks so you're not your server-side teams, which if you don't have a team, then you don't have to staff, hire, and maintain and manage a team. It's going to make it super affordable. Or if you do have a team, they could be doing something else that they're chartered to do as opposed to doing things that they don't

have to do because it's already available and you're not finding your customers who don't find value in you simply doing what is already there. That's the idea. So with that I think I'm going to end this video. Hope it makes sense. If you have any questions definitely let us know. We are more than happy to help you out. Thanks a lot. Bye bye.