Posted in: API (/spaces/127/sandpiper/forums/5044/api)

Security and Permissions

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/26/2019

Access

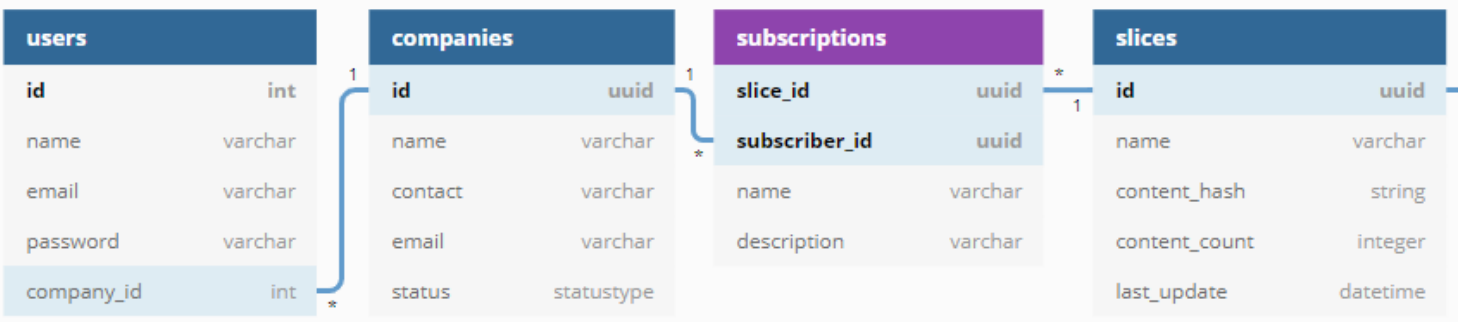
The API provides a way to add users. (As a side question, do we need a "sign-up" process, or should we make the Sandpiper admin add users?)

Authentication

The API uses username/password authentication for user login. It also provides a way to change password and other user information (e.g. email address). The API provides a way to recover (or reset) a forgotten password. Token-based authentication (JWT) is used for API access once the user is logged in. The token includes user ID and **access role** (see below).

Companies

Users belong to a Company (which must be supplied at user creation). Companies subscribe to Slices.



Roll Based Access Control

Each user is assigned a "roll". These rolls allow different levels of access. The "admin" roll is allowed full CRUD capabilities for all data elements.

We should probably restrict access to subscription maintenance. A basic "user access" should probably only allow sync operations. **What other rolls might we need?**

I can see the following access levels (with one level having all rights below):

Access Level	Rights
SuperAdmin	Complete access. Can add Admins.
Admin	Complete access except adding Admins.
Company	Can maintain users and subscriptions within that company.
Subscription	Can maintain subscriptions for a company.
User	Can initiate a sync for a company.

I've added the "Company" and "Subscription" levels for discussion purposes. **Question:** Do we need these levels or should the Sandpiper admin simply maintain all of this? I know this gets into the "contract" concept, so maybe that answers the question (in which case we probably don't need these levels). It would also simplify the implementation somewhat by not having them.

Database Level Permissions

The users and permissions mentioned above are all "virtual" in that they are maintained and enforced by the application. There is only one OS and RDBMS user used for application access.

access-rights (/spaces/127/sandpiper/searchresults?keyword=%22access-rights%22&searchtags=1)
authentiction (/spaces/127/sandpiper/searchresults?keyword=%22authentiction%22&searchtags=1)
jwt (/spaces/127/sandpiper/searchresults?keyword=%22jwt%22&searchtags=1)
login (/spaces/127/sandpiper/searchresults?keyword=%22login%22&searchtags=1)
password-reset (/spaces/127/sandpiper/searchresults?keyword=%22password-reset%22&searchtags=1)
rbac (/spaces/127/sandpiper/searchresults?keyword=%22rbac%22&searchtags=1)

Like

Reply (/forums/post?tid=5417&ReplyPostID=5418&SpaceID=127)

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
1/6/2020

> **On 12/26/2019 11:51 AM, Doug Winsby said:**
>
> I've added the "Company" and "Subscription" levels for discussion purposes. **Question:** Do we need these levels or should the Sandpiper admin simply maintain all of this? I know this gets into the "contract" concept, so maybe that answers the question (in which case we probably don't need these levels). It would also simplify the implementation somewhat by not having them.

An interesting quandary. I think there are two kinds of identity here: node and user.

For machine-machine transactions, the nodes themselves actually authenticate and pair with one another -- this identity needs to be provisioned on the node instance only by storing the data in buckets unique to the foreign node. Remember that no node should ever be able to write to or read from another node's snapshot pools. So there's almost no variation in security levels needed there -- every node can synchronize only its own data, no more or less.

For machine-human transactions and also within a node itself, that's where you might have users and roles.. Users of a Sandpiper instance can impact the core product data of the company, so there, the levels you've defined start to make sense.

I'd suggest you could get away with just Super, Admin, and User. Super and Admin are what you said, User would be able to initiate syncs etc.

Like

Reply (/forums/post?tid=5417&ReplyPostID=5462&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
1/6/2020

It would simplify things to do away with security levels. I'll see about removing what we can.

I still want to authenticate machine-to-machine links, though. The alternative is a simple "API Key" (just a shared UUID) that grants access and (most likely) indicates who is on the other end.

Like

Reply (/forums/post?tid=5417&ReplyPostID=5464&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
1/6/2020

> **On 1/6/2020 9:52 AM, Doug Winsby said:**
>
> I still want to authenticate machine-to-machine links, though. The alternative is a simple "API Key" (just a shared UUID) that grants access and (most likely) indicates who is on the other end.

I agree, the link should be authenticated. I was more saying that the authentication is for the node as an entity rather than users within it; the node itself is the one synchronizing its data.

Like

Reply (/forums/post?tid=5417&ReplyPostID=5468&SpaceID=127) Answer