(/) 🔍 (/searchresults) 🔔 (/myaccount/notifications)

Sandpiper (/spaces/127/sandpiper) ▸ Discussions (...
Private Space · Technology Standards (/spaces/9/technology-standards/technologyhome)
(/spaces/127/sandpiper)

Activity Stream (/spaces/127/sandpiper/?act=1) People (/spaces/127/sandpiper/people) Content ▾

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

Interaction Diagram

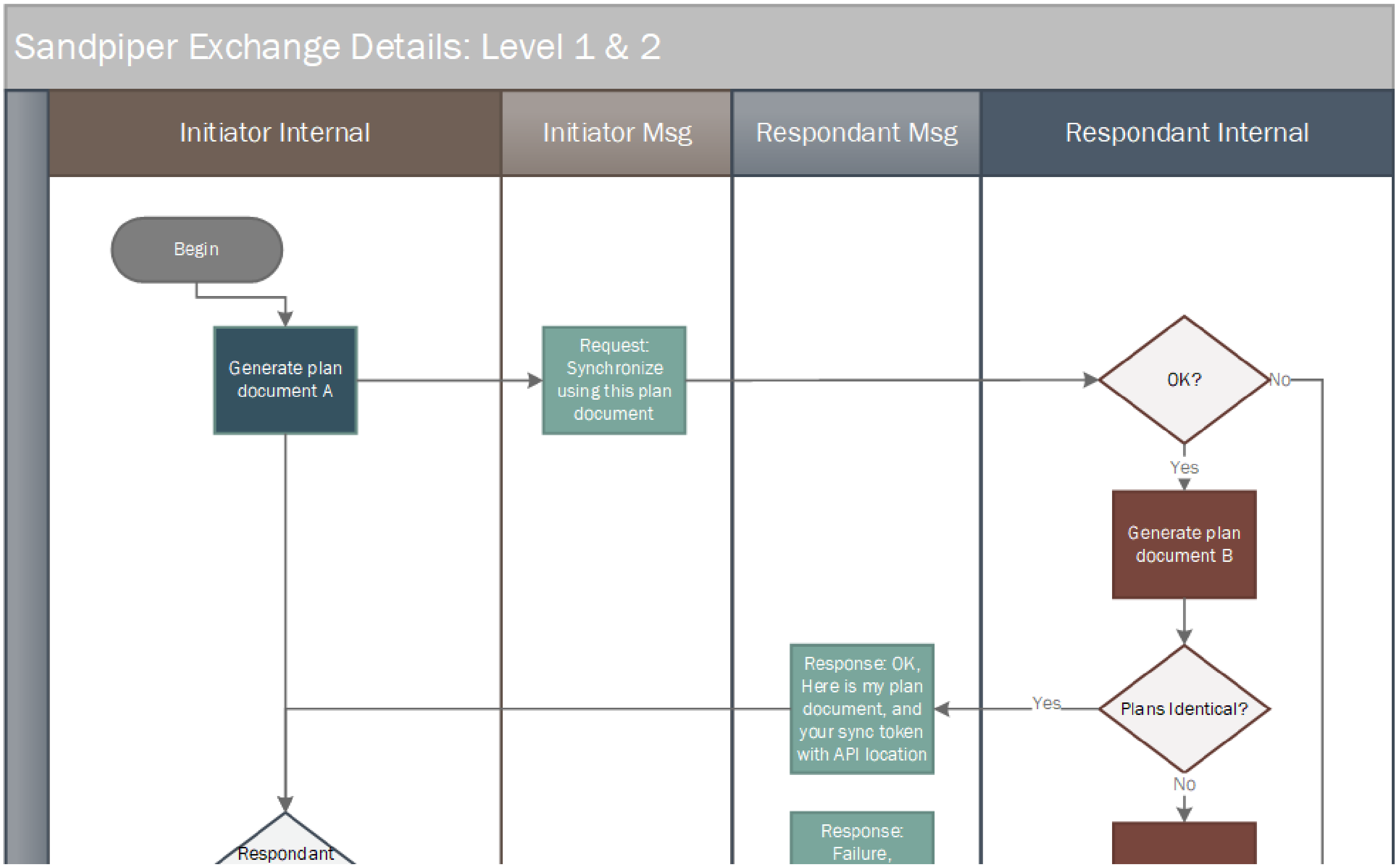
✉ Unsubscribe from this discussion

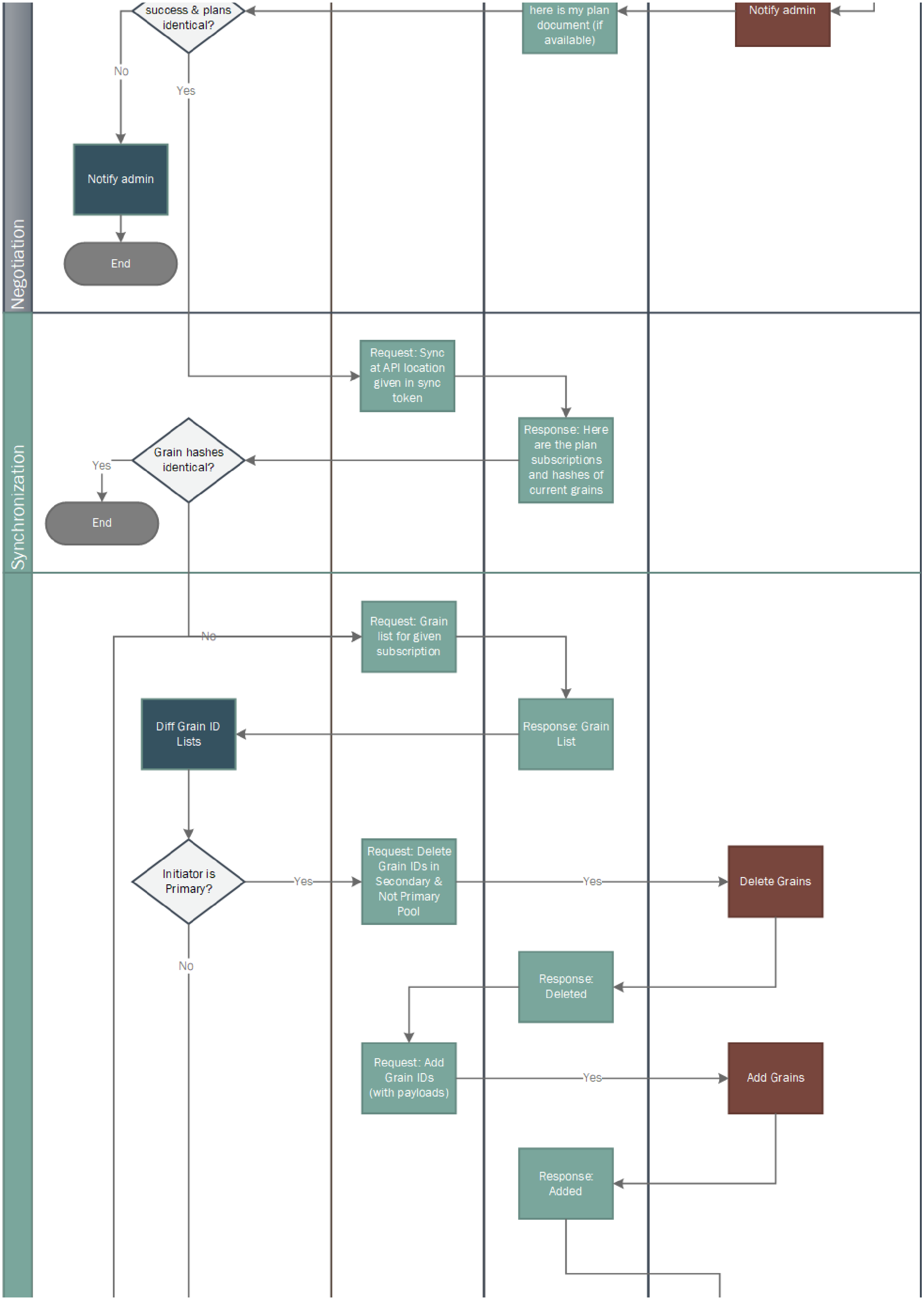
Subscribe to RSS (../../../../../../spaces-cf/forums/rss-space-posts.ashx?spaceID=127&topicID=6204&forumID=5044&key=rrer%2B1xDo%2BlxpC7jBRbM5w%3D%3D)

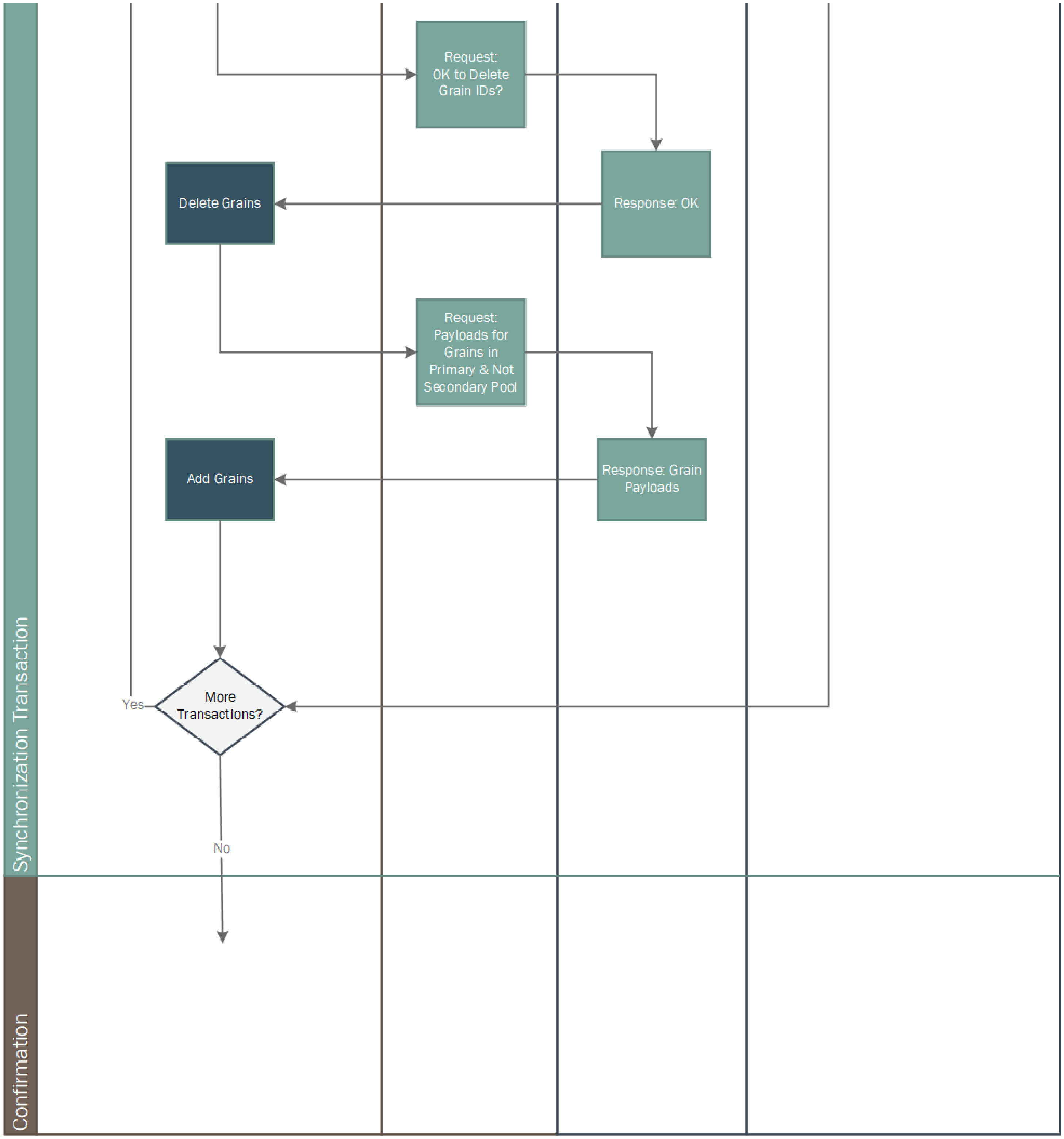
Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
12/4/2020

Hello, All,

This is the interaction diagram I've been prototyping with Luke for the RESTful exchange method. Work in progress!




Negotiation
success & plans identical?
here is my plan document (if available)
Notify admin
No
Yes
Notify admin
End
Synchronization
Request: Sync at API location given in sync token
Response: Here are the plan subscriptions and hashes of current grains
Grain hashes identical?
Yes
End
No
Request: Grain list for given subscription
Response: Grain List
Diff Grain ID Lists
Initiator is Primary?
Yes
Request: Delete Grain IDs in Secondary & Not Primary Pool
Yes
Delete Grains
No
Response: Deleted
Request: Add Grain IDs (with payloads)
Yes
Add Grains
Response: Added


Request: OK to Delete Grain IDs?
Response: OK
Delete Grains
Request: Payloads for Grains in Primary & Not Secondary Pool
Response: Grain Payloads
Add Grains
More Transactions?
Yes
No
Synchronization Transaction
Confirmation

👍 1 Like

Reply (/forums/post?tid=6204&ReplyPostID=6205&SpaceID=127)

Luke Smith (https://autocare.communifire.com/people/autopartsource)
12/16/2020

⋮

Krister, great work! This represents our initial attempt at plan-exchange in a mechanized way. I'll start the explanation here so we can get the debate juices flowing.

The key (new) concept here is that the plandocument is included in the initial "login" post action - along with the username and password. The document is xml or json bas64encoded as a blob of text. The authentication is done by the server with all three data elements. In your diagram, the first action (*Request: Synchronize using this plan document*) is more specifically: "*Request: Authenticate using this user, password and plandocument*". These three elements are conveyed in the body of the POST like this:

```
{
 "username": "wcoyote",
 "password":"m33pMeeP",
 "plandocument":"eyJwbGFuIjoiYzFiMWU1NTctZ..."
}
```

The response to a valid user/password is one of the following:

1) *I like you and your plandocument. Here is my version of the plandocument for your approval and (in case you like my version of the plan) here also is a JWT that will give you access to interacting with my content over the next 15 minutes.*

--- or ---

2) *I like you, but your plandocument has a problem. Here is my version of the plandocument for your inspection and recording keeping. Here also is your limited-access JWT for read-only, non-payload transactions over the next 15 minutes (assuming you want to window-shop a bit).*

The rationale is that the business relationship can be verified/quantified in both directions in a single API request. If the initiator (http client) is happy, it comes back moments later to interact with the appropriate API routes. If the respondent (http server) is happy, it issues a JWT that will allow access at the appropriate API routes moments later.

There is another new concept: The planID and a route access list (ACL) are encoded in the JWT. This way, every API service method is plan-aware and permissions-aware via the JWT. Here's how I think the JWT payload should look:

```
{
 "c":"efd926c8-3150-19be-8be1-64d2b1d6ed1e",
 "id":13,
 "u":"wcoyote",
 "exp":1607564050,
 "plan":"c1b1e557-f893-f2d0-6d75-1a0aeb3d520e",
 "resources":"activity,slices,grains,sync"
}
```

The plans are exchanged at authentication, and the planID is carried forward into the subsequent API calls by way of the JWT payload.

Like

Reply (/forums/post?tid=6204&ReplyPostID=6316&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/16/2020

Slight correction in the diagram. There are no **grain hashes**, only **slice hashes**. The gain UUID is what determines equality at the grain level. Grains are immutable and so the UUID always represents the grain contents. The slice hash also represents the slice metadata.

hashes (/spaces/127/sandpiper/searchresults?keyword=%22hashes%22&searchtags=1)

Like

Reply (/forums/post?tid=6204&ReplyPostID=6317&SpaceID=127) Answer

Luke Smith (https://autocare.communifire.com/people/autopartsource)
12/17/2020

Good observation, Doug.

Also, the box above that one should be changed from: *“Request: Sync at API location given in sync token”* to *“Request: List of subscriptions for our plan”*

I’m thinking the vertical labels along the left edge should be “Authentication”, “Evaluation” and “Synchronization”. The "Confirmation" section may not be unnecessary because the individual grain transactions are inherently confirmation-oriented.

Like

Reply (/forums/post?tid=6204&ReplyPostID=6318&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/18/2020

Still trying to digest the "plan" concept.

In the meantime, I thought it might be interesting to see how I implemented the sync by showing some of the code comments. These comments are taken from this file:

https://github.com/sandpiper-framework/sandpiper/blob/master/pkg/api/sync/sync.go (https://github.com/sandpiper-framework/sandpiper/blob/master/pkg/api/sync/sync.go)

> The Primary adds subscriptions (slices assigned to companies) and the Secondary asks for those assigned to them. This means that the Secondary (who currently initiates the sync) begins a sync session by asking for their subscriptions. It then adds any new ones to their local database.
>
> There is also an "active" subscription flag that can be changed (on either side). If disabled on the Primary, it will update the Secondary and log the activity. If enabled on the Primary, it will not change the Secondary. The active flag on the Secondary controls if it tries to sync that subscription, but changes are not propagated to the Primary. So, all of this means that the Primary controls what can be synced, but the Secondary can turn the sync off.
>
> The sync process will also observe the "active" company flag (on both sides) and the "allow_sync" flag which is false while the slice is being updated (on the Primary).

These comments describe the main code flow (in the Start function which starts a sync for a designated primary):

// **Start** sends a sync request to a primary sandpiper server from our secondary server
// log activity even if early exit
// must be a secondary server to start the sync
// must be a local admin to start the sync
// get company information for the primary server we're syncing
// connect to the primary server using their api-key (saving token)
// get our subscriptions (with slices) from the primary server
// get local subscriptions (with slices) as a receiver for this primary company
// sync all active subscriptions

Here are function comments from the above main logic:

// **syncSubscriptions** makes sure our local subscriptions match primary ones. If we don't
// have a subscription, add it locally. If disabled on the Primary, disable it on the
// Secondary and log the activity. If enabled on the Primary but not on the secondary,
// do not make any changes. Perform a grain sync on all unlocked active slices.

// run through primary (remote) subs and add to ours (local) if missing
// (the slice for a new subscription is also added, but not the slice metadata, which
// is added during the sync process)

// **syncSlice** does the actual work of looking for changes and doing the update.
// surround the sync with a begin/finalize update of the slice row to approximate a transaction
// and log results (and errors) to the activity table
// get remote grain list of ids
// get local grain list of ids
// determine local changes required to make local grains match remote grains
// remove obsolete grains (if any)
// add new grains
// replace local slice metadata with remote's
// update ContentHash, ContentCount & ContentDate and verify our own hash against remote's

Like

Reply (/forums/post?tid=6204&ReplyPostID=6320&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/19/2020

My point is that I feel the "plan" should simply be made up of the subscriptions and slice meta-data, and be maintained by the primary.

The sync process makes sure they agree on the plan (by duplicating that information on the secondary). If we want an api to extract the "plan" to a pdf or Excel for human consumption, we can easily do that.

It just adds a lot of complexity, in my opinion.

Like

Reply (/forums/post?tid=6204&ReplyPostID=6321&SpaceID=127) Answer

Luke Smith (https://autocare.communifire.com/people/autopartsource)
12/19/2020

Exchanging a standardized (strict xsd) plan codification at login is to facilitate a continual bi-directional re-affirmation of the human business agreement. I think the plan should originate from the primary on initial enrollment of the relationship (via API call). It would be up to the PIMs at either end to respect dictates of the plan.

We all agree that the Content should be dictatorially ruled by the author (primary). However, we recognize that the receiver (secondary) must have a veto option that sounds alarm bells in the primary. The "*redress of grievances*" concept in the first amendment comes to mind.

Like

Reply (/forums/post?tid=6204&ReplyPostID=6322&SpaceID=127)   Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/20/2020

This (or similar) is how I see the **sign-up process**:

1. Receiver goes to Primary public URL (sandpiper.bestbrakes.com) to request access.

2. Receiver requests access to sandpiper by entering their contact information. An email is sent with the request to the Primary admin with this contact information.

3. A phone call is made to vet the request and determine what subscriptions are needed.

4. Primary admin uses the admin web UI to create the company and user (without password). A "Send Invite" button sends this information to the requesting user via email with a special link.

5. The receiver gets the email and presses the special link taking them to a "reset password" work flow (they could also go to the web ui and press the reset password link because it is the same routine).

6. When they login, they can: (a) view the plan, (b) change profile info, (c) get a new sync apikey, (d) manually download individual grains assigned to them (via the browser).

7. They need to copy/paste this sync apikey into the correct customer record (e.g. Best Brakes) on their own sandpiper database (using the provided admin screens). I don't see any easier way of doing this because we don't save login credentials for the primary servers we sync with.

8. Any change made to a subscription definition (on the Primary), creates an email with that change sent to all receivers of that subscription.

Note that the receiver doesn't need to enter anything other than the apikey to allow syncs. Every sync also pulls subscriptions (the plan). Any requested changes to the Plan are made via email to the Primary contact.

Note that a primary could initiate the sign-up process starting at #4 without the initial request.

This method seems logical (and greatly simplified) to me.

Whatever we do, we need to imagine Epicor (for example) doing it hundreds of times to get everyone setup.

👍 Like

Reply (/forums/post?tid=6204&ReplyPostID=6323&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
12/21/2020

If I could restate the discussion, it might help me make sure I'm reading the issues right. I think we're talking about a few things, and it'll help to tackle one at a time. I'll save my thoughts on them until I've got the summary correct, so that you guys can correct any errors in understanding before I go too far down a rabbit hole that turns out to have been in my mind only.. Let me know how I did.

1. We have new terms to use when we need to refer to actors in terms of their network relationship rather than their data relationship
   1. **Initiator:** The actor making first contact
   2. **Respondent:** The actor accepting first contact
2. We have two methods of communication proposed
   1. **Persistent Connection / Peer-to-peer:** Nodes connect via WebSockets to create a persistent connection over which commands are executed.
   2. **REST / Initiator-Respondent:** Nodes connect using an intermittent HTTP RESTful API to execute actions
3. We have two methods of authentication proposed

    1. **API Key:** Actors share a manually-generated and manually-revokable unique secret, generated external to the Sandpiper database
    2. **Login:** Actors use a classic username / password login paradigm that is stored in both Sandpiper databases

4. We need to discuss the best way to initiate and modify an agreement, as our understanding of this concept was incomplete when we first developed solutions. Concerns to address:
    1. What changes, if any, are allowable without needing to ratify the new state of the agreement?
    2. The process needs to be simple enough to allow hundreds or thousands of relationships to be handled without too much burden
    3. What changes can be automated through the API, and what require a human being's involvement?

Like

Reply (/forums/post?tid=6204&ReplyPostID=6324&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/21/2020

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson) This is great. Defining terms and use cases is exactly what we need to do.

The one clarification I would add (because it is at the root of why we selected an apikey), is that all user passwords are stored encrypted. This means that plain-text passwords must be manually entered (and are not stored anywhere). So the dilemma was how to automate the sync (with all trading partners) without having to enter credentials manually.

Like

Reply (/forums/post?tid=6204&ReplyPostID=6325&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
12/22/2020

I think I'd like to open up a few new threads to handle these separately.. I imagine we have a lot to discuss, and we get mixed around trying to handle multiple topics in a single thread.

I'm focused on the XSD this morning but this afternoon I'll post some new messages where we can lay out the pieces and feel where the boundaries are, how to approach them best, etc..

Thank you as always for the insights and expertise!

👍 Like

Reply (/forums/post?tid=6204&ReplyPostID=6326&SpaceID=127)   Answer

Page 1 of 1 (11 items)