



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)

Contract / Plan

✉ Unsubscribe from this discussion

📡 Subscribe to RSS (../..../..../spaces-cf/forums/rss-space-posts.ashx?

spaceID=127&topicID=5057&forumID=5044&key=rrer%2B1xDo%2BlxpC7jBRbM5w%3D%3D)



Krister Kittelson (<https://autocare.communifire.com/people/krister-kittelson>)



8/16/2019

The context and agreement to operate is called the Plan. It's the contract between the upstream and downstream, which I'll still call origin and branch for the moment..

I'm writing example XML files to lay out how I think we should organize this information, but to get things down I'll go briefly into my thoughts here.

The origin and branch have their own elements under the root <Sandpiper> element, which the individual nodes fill out with their own information. The origin can only fill the origin element, the branch can only fill the branch.

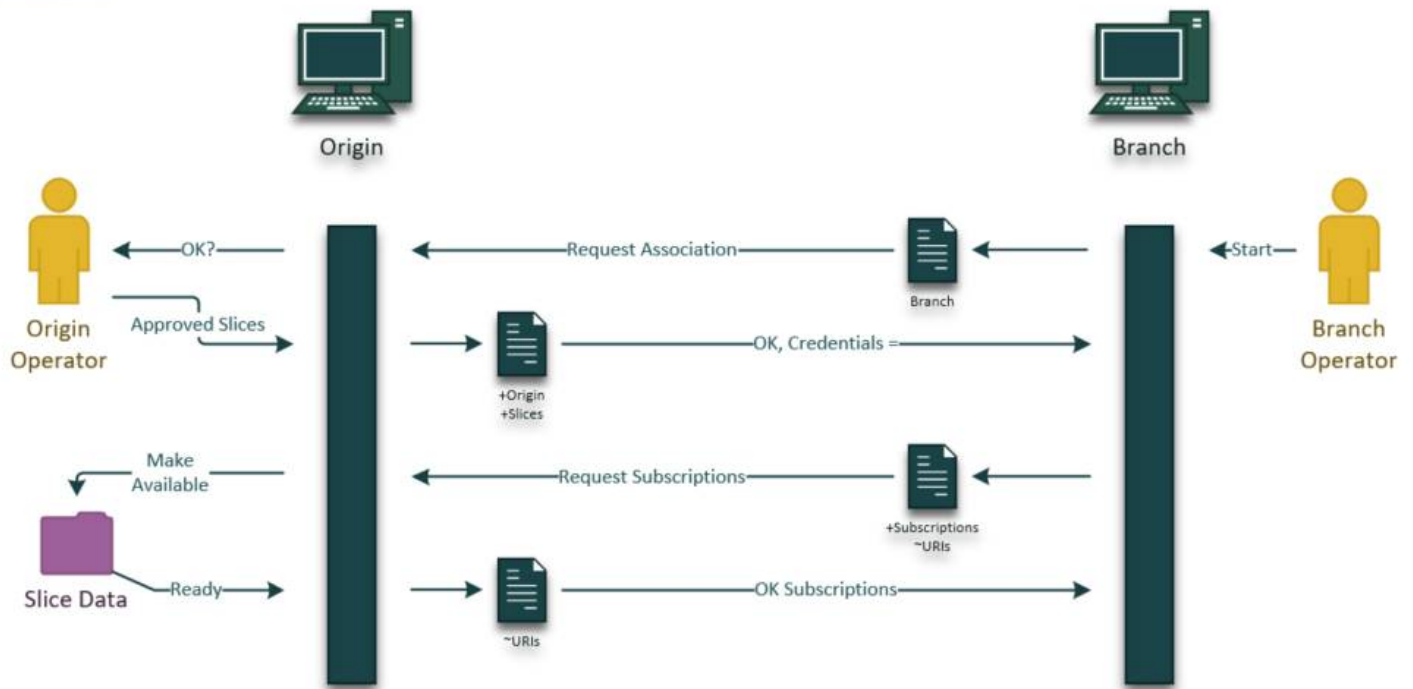
Both have <Organization> elements that allow defining things like contacts and links to industry standards like the AAIABrandID.

Within the origin is the <Slices> element, which allows defining data sets and slicing those datasets further with subslices. These link to other standards at several levels and also specify where the information can be accessed, how often, and by what method (including "API" as a method). This is where it gets fun and where there's a lot to write, so I'll just leave it there as a placeholder.

Within the branch is the <Subscriptions> element, where the branch states its intent to subscribe to a slice by its ID, how often it wants to get the data, and the method it will use to subscribe. This method can be "API" indicating the desire to engage in an API transaction for that slice.

So if you remember this image from my email a few weeks ago, this back-and-forth signing indicates explicit agreement to the terms, or rejection of the terms (which terminates the process with an error):

Association



👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5058&SpaceID=127)



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



:

8/16/2019

I think it would **greatly simplify the design** if we did not let a branch (i.e. secondary) request an association. The origin (primary) must approve it anyway, so why not just let them set it up where appropriate (probably when the salesperson comes in and says, "yippee, we got this new business").

Furthermore, it should be the "primary" that changes or terminates the association (when the salesperson says, "we didn't want that business anyway").

A RESTful design uses a "client-server" architecture (i.e. a stateless request-response approach). In Sandpiper, I see the primary as the "server" and the secondary as the "client" in this relationship.

In order for this to work, we need to define user "roles" in the API. I propose we need just two: "admin" and "user". The "admin" role allows complete control over all the "resources". The "user" role represents the normal "secondary" (client) access.

Also, I think of the hierarchy you show slightly differently. Instead of:

Associations -> Slices -> Subscriptions

I see it as:

Subscribers -> Subscriptions -> Slices

I feel this more clearly represents the one-way relationship.

👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5059&SpaceID=127)

Answer



Luke Smith (<https://autocare.communifire.com/people/autopartsource>)



8/17/2019

Doug, how does the marriage and divorce process play out in the case of a retailer and a manufacturer - when the retailer is the one hosting the API service. Let's use real-world actors and products: Brake Parts, inc, Raybestos and O'Reilly's.

👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5060&SpaceID=127)

Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



8/17/2019

I think this is maybe where our concepts differ.

I don't see the retailer in control of the relationship. Nothing says I must do business with O'Reilly just because they say so. It's up to the supplier to setup that relationship because they're the owner of the data that O'Reilly will be using to sell their products.

Also, I don't think the configuration setup needs to be machine-to-machine. I see a browser-based admin console for that purpose (with a human entering data). Basic CRUD operations on "Subscribers", "Subscriptions" and "Slices".

So in the case of O'Reilly, it's not clear to me that they would ever "host" the api (at least not as primary).

👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5061&SpaceID=127)

Answer



Luke Smith (<https://autocare.communifire.com/people/autopartsource>)



8/17/2019

In the specific case of O'Reilly's (and all retailers and e-cats), they have a thousand vendors who are the primaries for the product data. O'Reilly's is the logical actor to stand-up the API service and allow the majority of their vendors to push content in as clients to the API service because they (O'Reilly's) have the IT resources. The **minority** of those thousand vendors would have the sophistication and resources to maintain a reliable API service. ***O'Reilly's would be secondary in all cases*** - whether their system had received content from the vendor by being a server or by reaching out to the vendor as a client.

👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5062&SpaceID=127)

Answer



Luke Smith (<https://autocare.communifire.com/people/autopartsource>)



8/17/2019

"Also, I don't think the configuration setup needs to be machine-to-machine."

I completely agree - And I think this is where Krister and I diverge. I envision that the machines don't talk until the humans have established the pairing through an out-of-band side channel. The first interaction between peers is a run-of-the-mill content exchange - not a marriage ceremony.

👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5063&SpaceID=127)

Answer



Luke Smith (<https://autocare.communifire.com/people/autopartsource>) 

8/17/2019



I do still think it is a valuable mental exercise to describe the contracts/plans in xml terms. It forces us to consider the structure of the relationship's attributes and therefore the structure and rules of the actual relationship - even if that structure is never pushed across the pipe or stored as xml.




Like

Reply (/forums/post?tid=5057&ReplyPostID=5064&SpaceID=127)

Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>) 

8/17/2019



*"The **minority** of those thousand vendors would have the sophistication and resources to maintain a reliable API service."*

That's where the "reference implementation" comes in! I envision a pre-configured container that runs either on-premises or on any cloud service. There's really no excuse anymore. The "admin console" I mentioned would also allow a simple browser-based way to manage the service without IT assistance.

And for those that really don't want to deal with it, there will most certainly be a vibrant marketplace to service them (similar to what GCommerce IDE did for the AutoCare IPO standard).

*"**O'Reilly's would be secondary in all cases** - whether their system had received content from the vendor by being a server or by reaching out to the vendor as a client."*

In my view a "secondary" could never be the "server", only a "client".

Which brings up the larger question on how a sync can be initiated. I see two possible options:

1. The Secondary (client) asks the Primary API service periodically. "Hey, what do you have for me?" (Standard "polling" architecture)
2. The Primary (server) sends a message to the Secondary (client) through the feedback API. "Hey, I have something for you." (Standard "messaging" or event-driven architecture)

I don't really like the second option, especially since we haven't really discussed a "feedback API" yet. It also means there must be a listening server on the client side (and I was hoping to avoid that).

One answer might be to use a simple message queuing service (e.g. <https://www.rabbitmq.com/> (<https://www.rabbitmq.com/>), or <http://activemq.apache.org/> (<http://activemq.apache.org/>), or <http://queues.io/> (<http://queues.io/>), (<http://activemq.apache.org/>)). I like this idea because it might handle the whole "your service was down, then my service was down..." problem.

👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5065&SpaceID=127)

Answer



Luke Smith (<https://autocare.communifire.com/people/autopartsource>)



8/17/2019

Making the server always primary simplifies the framework. I think one of the large institutional receivers needs to weigh-in here. You and I are probably too narrow-minded to make that fundamental design choice. We're getting off-topic here. Let's take this fundamental topology discussion back to the other thread.

👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5066&SpaceID=127)

Answer



Krister Kittelson (<https://autocare.communifire.com/people/krister-kittelson>)



8/19/2019

It helps to split this into two different things: the data and the relationship.

The data is more like you are describing, Doug, a one-way-street, an owner and a user. I'd like to talk more about that in another post so I don't confuse myself.

The relationship, however, is a two-way action, and provably so. Either party's refusal to participate ends the relationship, because if I refuse to send the data or you refuse to accept it, we are no longer cooperating. Therefore it should be dealt with as two independent and equal actors who are agreeing that they will enter into an unequal data state. They still remain independent and capable of leaving at any time.

The relationship portion of Sandpiper is one piece that is truly an attempt to solidify and standardize what humans are already doing (and thus remove all the ambiguity and deviant end-states). We do need to limit the scope to make it as implementable and deterministic as possible though!

The concept of statelessness is true of the protocol, but that's why the document itself is being revised and sent between the two parties. The document itself tracks the state of the interaction. Therefore the API just has to handle "Contact, Send Document".

So to help limit the scope, perhaps the initial contract can explicitly indicate what kind of relationship the initiator is attempting to establish. The other end can agree, and things keep going forward, or disagree, and things stop. Between these steps is probably a human being reviewing them:

1. Actor A starts up the talk, saying, "Hey, I want to be a (primary | secondary) to you. Here's who I am."
2. Actor B says, "Here's who I am when I'm (primary | secondary) and what I could accept in a relationship between us."

The primary | secondary can be established in that initial contact by having an empty element for the other half of the relationship. Saying basically, "Heyyyy, write your details down here, wouldja?"

👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5071&SpaceID=127)

Answer



Doug Winsby (<https://autocare.communifire.com/people/dougwinsby>)



8/19/2019

I can see how a "sign-up" process (a formalized way for a secondary to request approval to sync certain product data) would be very helpful.

I'm still struggling with the reverse of that, however. Wouldn't the reverse just be the primary setting up the sign-up without the initial request from the secondary? It would then be up to the secondary to use the credentials to do the sync or not (i.e. to accept the proposed businesses relationship or not).

👍 Like

Reply (/forums/post?tid=5057&ReplyPostID=5072&SpaceID=127)

Answer

Copyright © 2021 Axero Solutions LLC.
Auto Care Association powered by Communifire™ Version 8.0.7789.8960

© 2021 - AUTO CARE ASSOCIATION (<http://autocare.org>) | LEGAL & PRIVACY STATEMENT
(<https://www.autocare.org/privacy-statement/>)