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

"Level 1" Design

Unsubscribe from this discussion
Subscribe to RSS (../../../../../../spaces-cf/forums/rss-space-posts.ashx?spaceID=127&topicID=5410&forumID=5044&key=rrer%2B1xDo%2BlxpC7jBRbM5w%3D%3D)

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/21/2019

The current thinking is that Sandpiper "**Level 1**" would support only **file-based** syncs. This includes:

payload-type	**key**	**level**	**description**
aces-file	none	1	full file
pies-file	none	1	full file
partspro-file	none	1	full file
asset-file	asset-id	1	full file (not meta-data)

Furthermore, we've determined that each object is assigned to just one "slice".

This brings up a basic question, however.

> Should we allow more than one file-based payload-type per slice? In other words, should we allow more than one "aces-file" to be assigned to a slice?

To answer this question properly, we need a very clear **definition of a slice**. Here is a portion of an earlier post I made that gave some possible "examples":

- All part numbers for a part-type using primary numbering (dynamic)

- All part numbers for a part-type using customer numbering (dynamic)
- A subset of a part-type filtered by sub-brand using customer numbering (dynamic)
- A subset of a part-type filtered by part-attribute (e.g. remanufactured) (dynamic)
- A list of part numbers using customer numbering (static)

If this is off the mark, or you can think of other examples, **please chime in**.

We know trading partners can "subscribe" to more than one slice. But that's only helpful when data-objects are fine-grained. File-based delivery is very coarse-grained.

All the slice examples listed above (except the last) seem to be at a lower level than normal file-based objects. I generally see "All part numbers for a product line" granularity.

Since files are usually course-grained, this seems to point towards the one file-type per slice concept. It would sure be easier to enforce a one-to-one relationship, but it feels overly restrictive.

If, however, we allow a many-to-one (file-object to slice) relationship, we'd need a way to identify which ones to replace. So we'd need a "key" of some type that defines uniqueness within a slice (object-id is unique to the universe). For ACES, it might be:

> ```
> aces-file, document-title
> ```

It would then be up to the "add file-object" API to accept these keys and remove the old file-object before adding the new one. Or at least to provide a way to query the object-id based on these keys so the delete could be performed first (and generate an error if trying to add a duplicate key).

If we allow a many-to-one relationship, we need to define the keys for each file-type. I don't like that dependency, especially since I'm not sure what the key would be for PartsPro (for example).

If we allow an "asset-file" payload-type, we certainly require a key of some sort (I like the PIES "P06" asset-id for this purpose since it already exists). But do we need keys for the other file types? That is the question.

Please provide feedback. This needs to be decided before we can move forward.

file-based (/spaces/127/sandpiper/searchresults?keyword=%22file-based%22&searchtags=1)
file-object (/spaces/127/sandpiper/searchresults?keyword=%22file-object%22&searchtags=1)
level1 (/spaces/127/sandpiper/searchresults?keyword=%22level1%22&searchtags=1)

Like

Reply (/forums/post?tid=5410&ReplyPostID=5411&SpaceID=127)

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/21/2019

A related question is if we should assign a "Level" to a `slice`?

Should we allow a mixture of file-based and lower-level objects in a slice, or should all objects in a slice have the same sandpiper-level?

slice-level (/spaces/127/sandpiper/searchresults?keyword=%22slice-level%22&searchtags=1)

Like

Reply (/forums/post?tid=5410&ReplyPostID=5412&SpaceID=127) Answer

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

I haven't received any feedback on the "related question" above, but it seems odd to mix `aces-file` objects with `aces-item` objects, for example. But we might want to mix `aces-item` objects with `pies-file` objects.

So we probably need some other way to limit payload-type combinations besides simply including an implementation level in the slice.

Like

Reply (/forums/post?tid=5410&ReplyPostID=5419&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
12/27/2019

I've been writing up the 0.8 documentation and am getting to the levels part this morning.. I should have some feedback for you later today! I wanted to make sure my thoughts were in order and I had a common point to come back to.

Like

Reply (/forums/post?tid=5410&ReplyPostID=5420&SpaceID=127)   Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
12/27/2019

> **On 12/21/2019 2:11 PM, Doug Winsby said:**
>
> The current thinking is that Sandpiper "**Level 1**" would support only **file-based** syncs.

I think more that level 1 performs "complete" synchronizations, delivered as files. However, this doesn't mean that the sources have to be one file, or multiple files. Sandpiper could take in one ACES file as its source, but define several slices within it, each of which would be delivered (regurgitated) as a standalone file. Thus if the slice defines a set of product lines, the source data would be split into those lines by the criteria defined in the slice, and delivered as a full XML / TXT / etc. file.

But these still must be the one file type per slice, you're right, because the Set is one file type. And a slice belongs to one set. It's a little restrictive though it greatly reduces complexity to do it this way...

I like the idea of defining slice-to-slice relationships but it may need to be in a different revision of Sandpiper. In the current approach, the onus is on the data owner to ensure that the slices are equivalent in scope, so that the same ACES, PIES data can be delivered at the exact same level. But since Level 1 is in large part about automating the way people do things today, I know that these files are often delivered with different scopes. For example, BPI currently delivers separate Raybestos friction ACES files to Epicor, but we deliver one Raybestos asset file, because of a system limitation.

So while I do like the idea of further functionality, I think for Level 1 the basic coarse mechanism is sufficient at the moment.

Now, that said, I had originally given some thought to having defined "scope templates" that would have a unique ID, belonging to the pool. The slice would identify its criteria only as this ID, and in this way identical scopes could be used throughout the pool, across many file types. It could be something neat for Sandpiper 1.1.

On the model for replacing full files: the only ID for a Level 1 transaction's scope should be the slice ID. I very much want the interaction logic to be simple at this level. Level 2 can get into defining grains and elements below, but Level 1 is pure and repeatable. Anything the secondary has corresponding to that unique slice ID should be dropped / archived and replaced with the new state of the data assigned to that slice ID.

Like

Reply (/forums/post?tid=5410&ReplyPostID=5421&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/27/2019

> "Sandpiper could take in one ACES file as its source, but define several slices within it, each of which would be delivered (regurgitated) as a standalone file. Thus if the slice defines a set of product lines, the source data would be split into those lines by the criteria defined in the slice, and delivered as a full XML / TXT / etc. file."

Are you saying Sandpiper would need to understand the Standards well enough to perform operations on the payloads? This appears to break the "black-box-payload" concept.

Like

Reply (/forums/post?tid=5410&ReplyPostID=5422&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
12/27/2019

> **On 12/27/2019 3:09 PM, Doug Winsby said:**
>
> Are you saying Sandpiper would need to understand the Standards well enough to perform operations on the payloads? This appears to break the "black-box-payload" concept.

You're right that the capability is not one I'd want to see handled in the Sandpiper framework. It's too broad for our goals here.

One way or another, the protocol and framework should not need to know anything about the contents of the data, only either to deliver it in total or how to work from a single unique ID to add and delete.

I was thinking more along the lines of the server implementation understanding the PIM data well enough to segment it, and pass it on via the black box payload (maybe caching it beforehand or as part of a publication routine). But for a reference implementation this doesn't make sense. It reminds me of the need for a C++ preprocessor -- the compiler is what we're building, the PIM either needs to feed us the ready-to-go files or something between the PIM and us needs to do it.

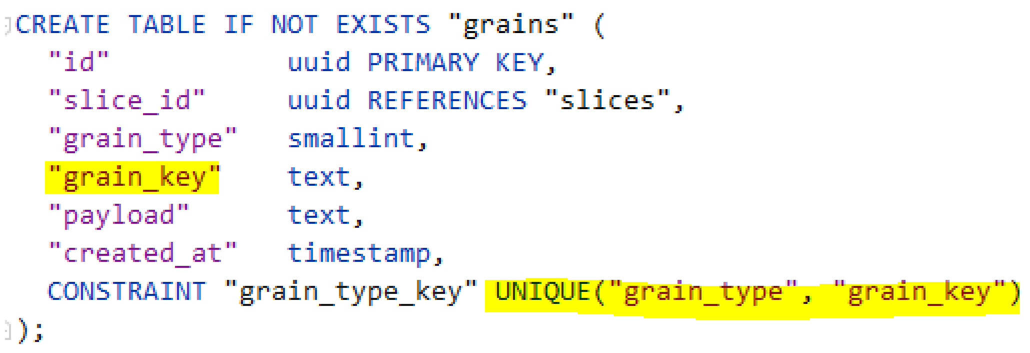
Like

Reply (/forums/post?tid=5410&ReplyPostID=5423&SpaceID=127)   Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
1/4/2020

It sounds like we need to allow multiple "aces-file" grains (for example) in a slice. We could leave this entirely up to the PIM to manage (deleting old ones and replacing them with new), but it makes sense to add some database constraints where we can.

With that in mind, the grain table has been changed to include a `grain-key` as shown below with a unique key on [grain-type, grain-key]. (Note that we've also changed the name of "payload-type" to "grain-type" to better reflect its meaning.)



```
CREATE TABLE IF NOT EXISTS "grains" (
  "id"           uuid PRIMARY KEY,
  "slice_id"     uuid REFERENCES "slices",
  "grain_type"   smallint,
  "grain_key"    text,
  "payload"      text,
  "created_at"   timestamp,
  CONSTRAINT "grain_type_key" UNIQUE("grain_type", "grain_key")
);
```

This allows us to add `asset_id` to identify "asset-file" grains, and any other identifier the PIM might need to uniquely identify other file-based grains. When we move to "item" based grains (in Level 3), this key would contain the item number (i.e. Part Number).

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

Like

Reply (/forums/post?tid=5410&ReplyPostID=5453&SpaceID=127)   Answer

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

> **On 1/4/2020 5:45 PM, Doug Winsby said:**
>
> It sounds like we need to allow multiple "aces-file" grains (for example) in a slice. We could leave this entirely up to the PIM to manage (deleting old ones and replacing them with new), but it makes sense to add some database constraints where we can.

It's an interesting thing, this Level 1.5-ish part number slicing. You've deftly revealed the consequences of a file-based transfer when part number subdivision is allowed. I wonder if we're overcomplicating and should revert Level 1 back to being full transfer only?

Still, these db schema modifications definitely make sense looking forward. I have made similar payload-key-tags in some of the db-side XML processing I do and it makes a huge difference in performance and integrity.

Like

Reply (/forums/post?tid=5410&ReplyPostID=5459&SpaceID=127) Answer

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

> **On 1/6/2020 9:25 AM, Krister Kittelson said:**
>
> I wonder if we're overcomplicating and should revert Level 1 back to being full transfer only?

I still see Level 1 as "full transfers". I guess I've always thought of it as just a fancy way to deliver what they are currently producing.

We could make a rule of one aces-file per slice, but we need to support sending many asset-files, so need the key for that case anyway.

Like

Reply (/forums/post?tid=5410&ReplyPostID=5461&SpaceID=127) Answer

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

> **On 1/6/2020 9:39 AM, Doug Winsby said:**
>
> We could make a rule of one aces-file per slice, but we need to support sending many asset-files, so need the key for that case anyway.

A good point.

Okay, so let's go forward with the idea that a grain in a level 1 transaction is going to be simple and full: actual files. If you have multiple ACES files, PIES files, Asset files, Spreadsheets, etc., you can send those as single grains within a slice.

I think we might also want to say that grain windowing as a sync instruction is not supported in Level 1. If a key criterion is added to a grain, it is treated as informational for use later in the processing chain, not as an instruction for full replacement.

I'll refine the documentation to note these if it sounds like we're on the same page!

Like

Reply (/forums/post?tid=5410&ReplyPostID=5463&SpaceID=127) Answer

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

> **On 1/6/2020 9:46 AM, Krister Kittelson said:**
>
> If a key criterion is added to a grain, it is treated as informational for use later in the processing chain, not as an instruction for full replacement.

Right. I intended the key only as a way for the PIM to identify a grain by something other than the grain-id (UUID). So it could say, for example, delete the "Brake Pad" aces-file from the "AAP-Brakes" slice because I need to update it.

Like

Reply (/forums/post?tid=5410&ReplyPostID=5465&SpaceID=127) Answer