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

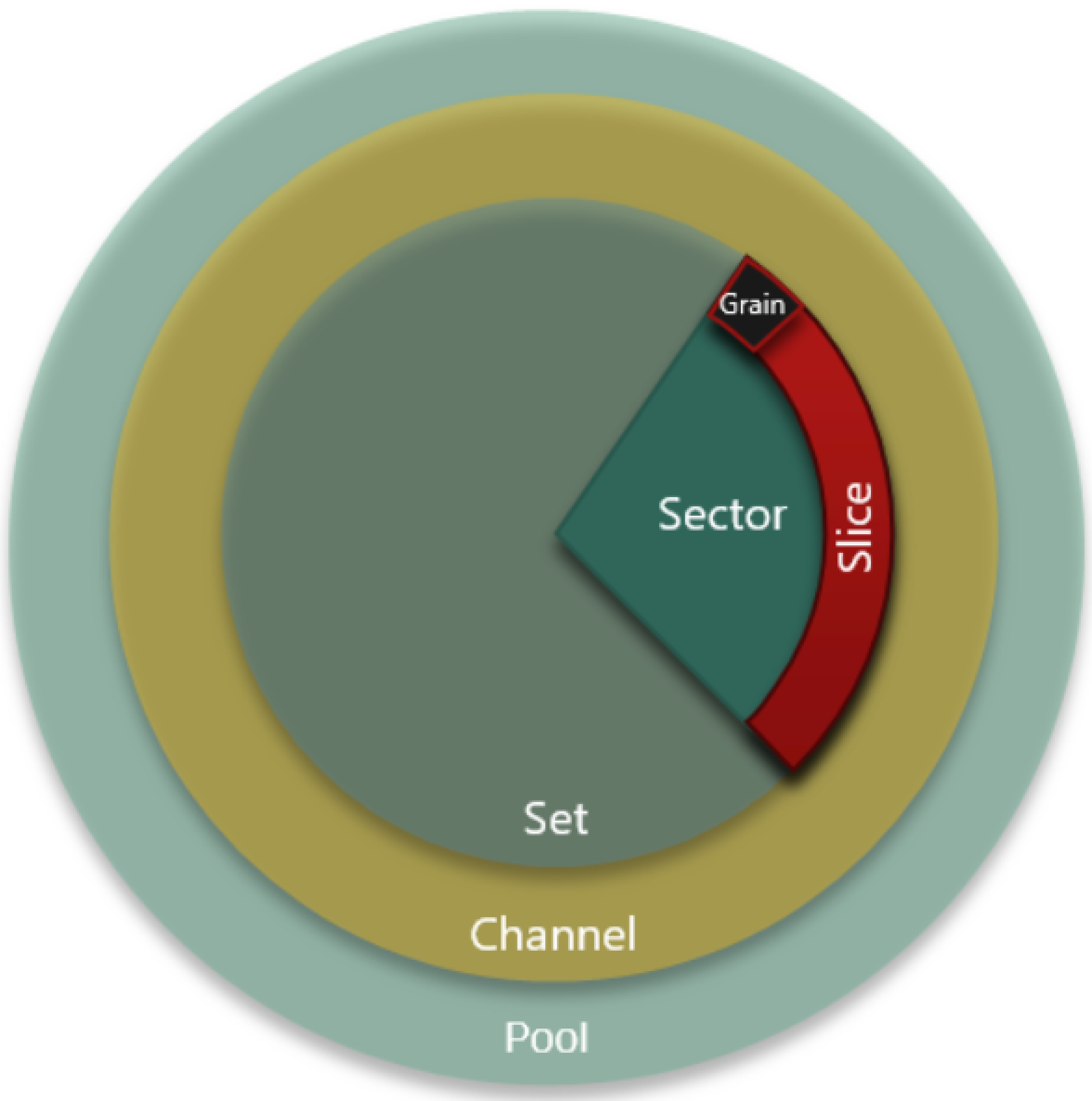
Proposal for Slice Aggregation & Terms Update

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
10/25/2019

Hello, All,

Based on the discussions we've had over the last few weeks, I've been thinking about how to aggregate slices and also clear up the terminology of a slice.

Doug proposed a higher level of organization within a pool that he called a channel. I like the idea and have made a slight tweak in terms to fit a model based on divisions of data -- how does this sound?



1. A Node is the individual instance of a Sandpiper system
2. A Node contains one canonical Pool and may store multiple snapshot Pools
    1. A Pool is the collation of all data belonging to one Sandpiper Node
3. A Pool contains one or more Channels
    1. A Channel represents one major format of data
        1. For example, ACES fitment, PIES product information, Excel part specifications
    2. A Channel can reference specific releases of related system databases, e.g. the VCdb version date
3. A Channel contains one or more Sets

   1. A Set represents data of a specific format, able to be processed and stored using the same chain of methods
      1. For example, the field names and dimensions are the same, for a flat file, or the Elements conform to the same schema, for XML, or the binary data is PNG format, for an image, etc.
      2. ACES v3.2, PartsPro, TecDoc, PIES v6.7, PIES v6.2...

5. A Set contains one or more Sectors
   1. A Sector is a general division of the Set
   2. A Sector uses common general or part-driven criteria (e.g. Branding, Part Type)
6. A Sector contains one or more Slices
   1. A Slice is a minor division of the Set
   2. A Slice is the most specific division element in the Plan; further divisions must take place in Level 2 interactions
7. A Slice can be operated upon using Grains
   1. A Grain is a very low level window inside a Slice, used to implement interaction-specific data changes

I'll post a concrete example next, but this is long enough for one post probably.

Like

Reply (/forums/post?tid=5279&ReplyPostID=5280&SpaceID=127)

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
10/31/2019

One of the most difficult tasks in good software design is coming up with a succinct Domain Model (https://www.scaledagileframework.com/domain-modeling/). In the case of Sandpiper, it becomes even more difficult because we're trying to model an entire industry (rather than just one company's requirements).

I like that we've narrowed down the vocabulary (and definitions) that we can all understand and agree on. But I can't help think that we're making this more complicated than necessary.

```
grain -> slice -> sector -> set -> channel (-> pool)
```

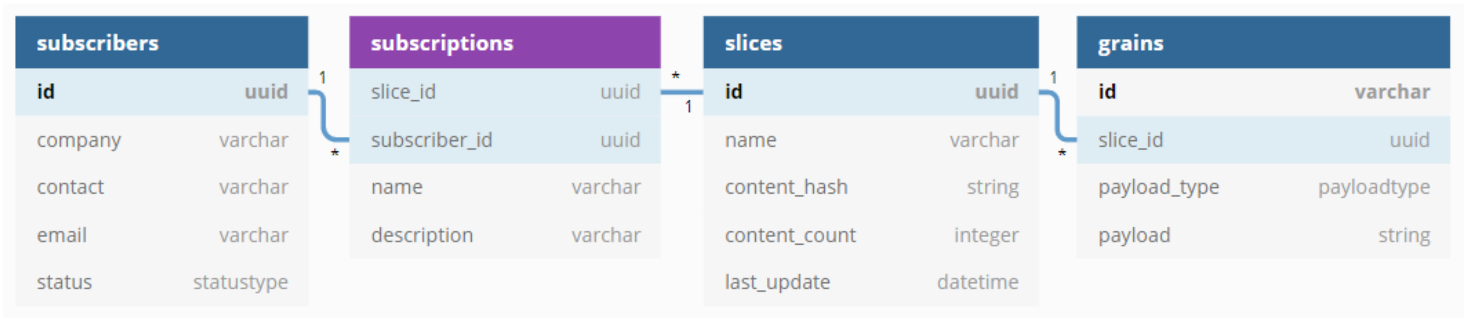
That's a lot of nesting. Can we accomplish the same thing with less levels? I was recently thinking this would take care of it:

```
grain -> slice -> channel
```

But now I feel the term "channel" might be confused with other meanings in our industry, and so be too restrictive (assuming the "green" data-model implementation here (https://autocare.communifire.com/spaces/127/sandpiper/forums/api/5271/data-model-er-diagram?postid=5273#5273)). Maybe something like the following is more general (and so flexible):

```
grain -> slice -> subscriptions
```

This data-model would look like the "purple" implementation.



subscribers	
id	uuid
company	varchar
contact	varchar
email	varchar
status	statustype

subscriptions	
slice_id	uuid
subscriber_id	uuid
name	varchar
description	varchar

slices	
id	uuid
name	varchar
content_hash	string
content_count	integer
last_update	datetime

grains	
id	varchar
slice_id	uuid
payload_type	payloadtype
payload	string

This would allow any company to "subscribe" to any "slice". The admin screens would make this easy to accomplish (provided `slices.name` values are sufficiently descriptive).

Does this accomplish what we need? Are there real examples available where this might fall short?

Like

Reply (/forums/post?tid=5279&ReplyPostID=5295&SpaceID=127)   Answer

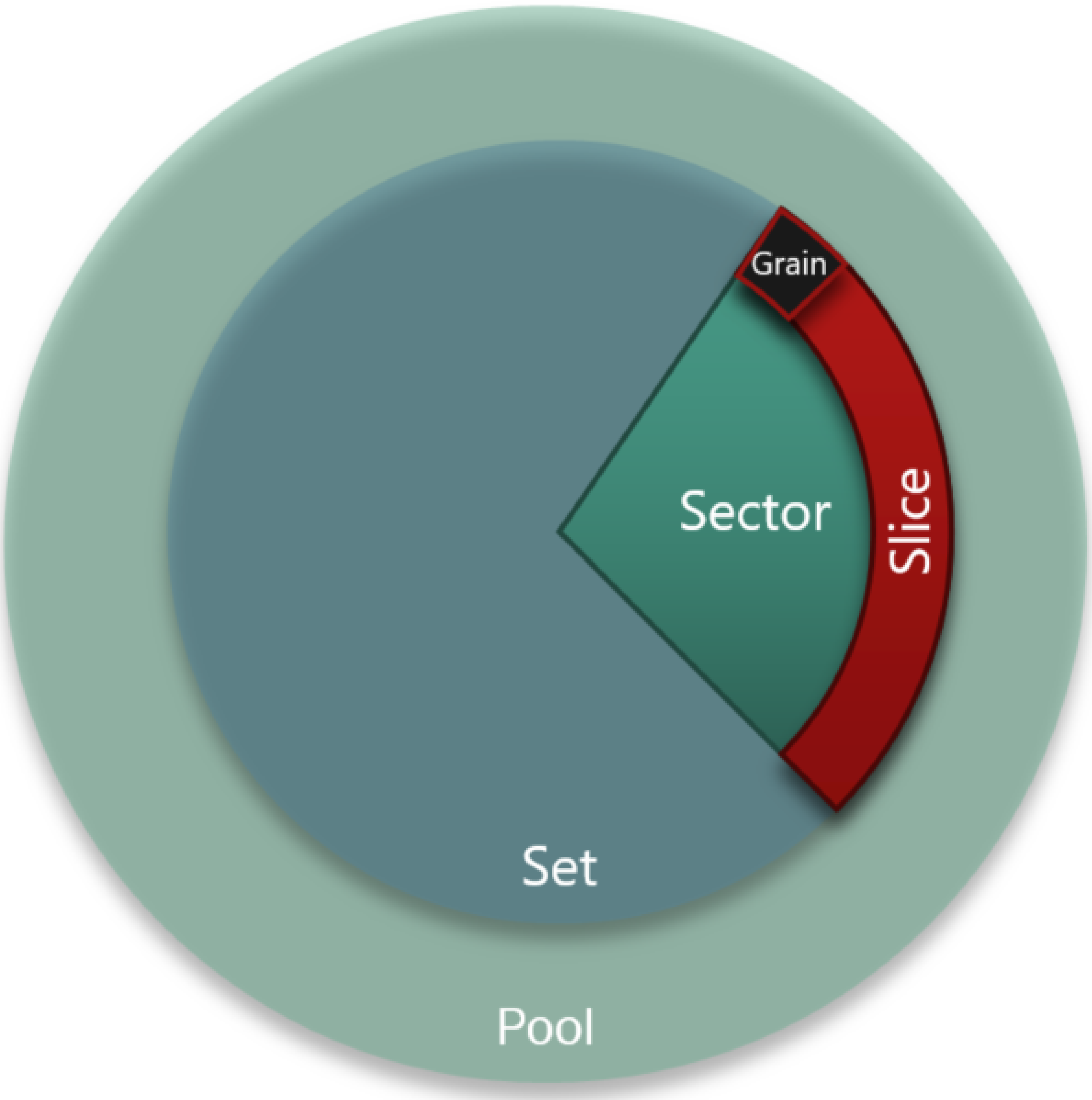
Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
10/31/2019

I think the subscription does exist, but it is an element between a secondary node and some part of the data in the primary node's pool. A party subscribes to the data, but the data is independent of the subscription. That way data categorization doesn't have to happen for every person who wants to access it -- only when they need that kind of special segmentation, and only when existing "standard" categorizations won't serve the need. This will keep people from having to strike a new classification every time someone wants to receive the data.

So I think this comes in as part of the plan, but the plan itself will have two sections: the primary and the secondary. The primary states what it offers, the secondary states what it wants, they both agree that this seems good and life goes either to the next level or humans manually proceed with ad-hoc downloads by hand.

Re: complexity, you're right that these could be going too far; I'm trying to find a compromise between an infinite hierarchy / nesting model, and a one-level flat model. I do think having a bit more ability than just the channel is necessary; the ultimate categorization of product should happen in the plan, so that the different needs (security, distribution, integration, etc) for different brands and part types can be handled. For example, XYZ can access Raybestos Rotors but not Friction, yet XYZ is allowed to access all Aimco products regardless of type.

I could, however, see removing one of these levels. The Channel:Set divide is one of the basic product information type (ACES fitment etc) and the actual format of that information (e.g. ACES 3.1). But in thinking about it with clearer perspective maybe it's OK to just have one level here, and specify the format and type at the same step. I had intended this to be especially for PIES or Excel, where you might need to provide multiple versions of the data, but really this split doesn't add much love to the world in the end.



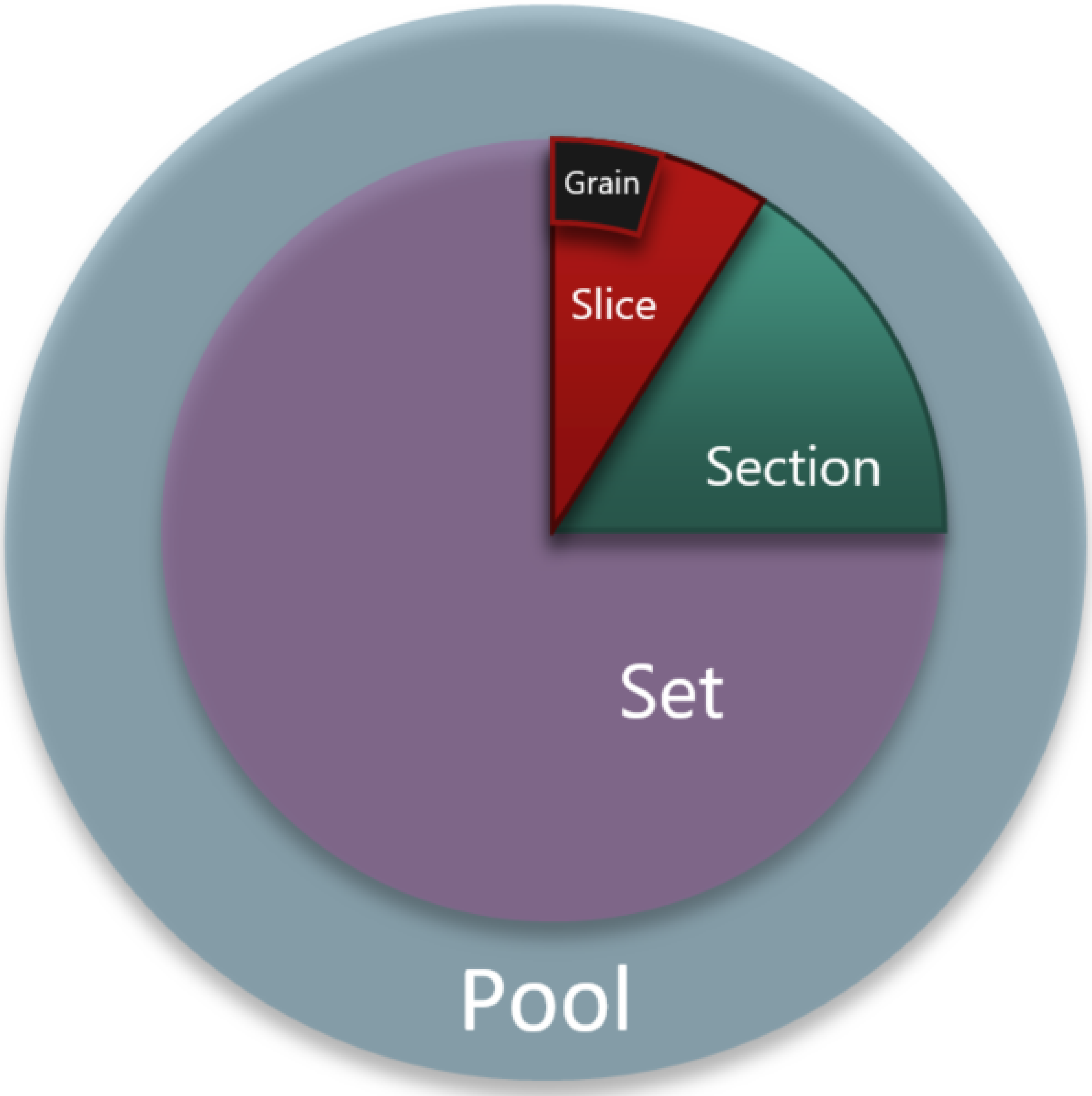
Where that would put us, then, is a little simpler: Pool->Set->Sector->Slice. Then the grains are chosen during interaction. The user would only care about three levels, really, with the set, sector, and slice. An actor would subscribe to any slice.

Like

Reply (/forums/post?tid=5279&ReplyPostID=5296&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
10/31/2019

You know what, after wibbling about with the visual I think I have a better one, and a friendlier name for Sector: Section. This way the slice retains the "Slice of Pie" idea:



Like

Reply (/forums/post?tid=5279&ReplyPostID=5298&SpaceID=127)   Answer

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

10/31/2019

By adding those higher levels (`sector` and `set`), I think maybe you're duplicating the responsibility of the "`slice`".

As I understand it, a slice cuts across traditional segmentation. This implies that we can do away with those segments. We're essentially pushing those definitions outside the scope of Sandpiper (which I like because it greatly simplifies things).

So, if XYZ wants Raybestos Rotors but not Friction, plus all of Aimco, the PIM must create one or more slices with just those part numbers (how it decides this is up to their implementation). Then the admin would assign those slices to XYZ's subscriptions.

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

Like

Reply (/forums/post?tid=5279&ReplyPostID=5299&SpaceID=127)   Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
10/31/2019

The way I thought of it, set definitely does not duplicate the segmentation; its pure purpose is to define the type of data contained within. Only one set of any given type would be allowed per pool -- one ACES 3.1 set, one ACES 3.0, one PIES 6.5, etc.

But for the section and slice, yes: the slice is the endpoint, a single reference of a grouping of data. What I'm hoping to do is make the actual grouping of data more rigorous, so that slices don't overlap -- so that they are truly and verifiably discrete elements of a larger whole. That way, you can know that by adding or removing an additional set, you're not interfering with pre-existing data; you can determine exactly what the nature of the transaction will be.

If a slice is allowed to define all of its own classification, there are fewer ways to enforce sameness of classification among slices. By putting the responsibility for declaring the type of division at least one level above a slice, you guarantee a common method of divisibility, yet reserve some flexibility to allow wildly different categories in your business to be defined separately.

For example, for a fictional company selling just brake friction and brake iron, with two brands of pads and only one of rotors, they may have this kind of division:

Sets

ACES 3.1

- *Section Type: Part Type Group*
- ***Sections***
- Brake Friction
  - *Slice Type: Brand*
  - **Slices**
  - ULTIMO

PERFORM

Brake Iron

Slice Type: Part Type

Slices

Rotors

Drums

But then they acquire a filters company that has dozens of subbrands. Rather than having to apply that subbrand strategy to their existing slices, they can accommodate the new section:

Sets

ACES 3.1

Section Type: Part Type Group

Sections

Brake Friction

Slice Type: Brand

Slices

ULTIMO

PERFORM

Brake Iron

Slice Type: Part Type

Slices

Rotors

Drums

Air Filters

Slice Type: SubBrand

Slices

Filtremo Performance

Filtremo Daily

Cabnator Plus

Oil Filters

Slice Type: SubBrand

Slices

Filtroil Gold

Filtroil Green

Filtroil Bronze

Like

Reply (/forums/post?tid=5279&ReplyPostID=5302&SpaceID=127)   Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
11/1/2019

> *"...one ACES 3.1 set, one ACES 3.0, one PIES 6.5, etc."*

I had conveniently forgotten about delivery versions! What a foobar! Just when I thought we were out of the woods with reference-versioning, we now need to deal with delivery-versioning too!

Like

Reply (/forums/post?tid=5279&ReplyPostID=5310&SpaceID=127)   Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
11/1/2019

> **On 11/1/2019 9:32 AM, Doug Winsby said:**
>
> Just when I thought we were out of the woods with reference-versioning, we now need to deal with delivery-versioning too!

It's woods all the way down! :)

Like

Reply (/forums/post?tid=5279&ReplyPostID=5311&SpaceID=127)   Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
11/1/2019

Answered

I feel the answer to this problem may be, once again, to push it back to the PIM. The driving concept behind the Pool (in my mind) was to remove dependencies, structure and state. It's basically a collection of "sycnable-objects". The more structure we have, the less flexible it becomes.

So, as with "reference-versioning", "delivery-versioning" should be "set" for a slice. "There can be only one (https://en.wikipedia.org/wiki/Highlander_(film))".

We already have "slice-metadata" defined, maybe we also need "grain-metadata". The PIM would need to know which type of grains to create (by looking at all the slice requirements), and then properly associate grains with slices. Conceptually, the meta-data might look like this:

> `slice delivery-versions: [{"pies-version","7.1"}, {"aces-version", "4.0"},...]`

> `grain delivery-version: {"aces-version", "3.2"}`

That `grain` could not be added to that `slice`.

Sandpiper Admin, which is used to assign slices to subscriptions, could also check the slice delivery-version to make sure a subscription doesn't mix them.

EDIT: Actually, if a `grain` is only associated with one `slice`, we don't strictly need delivery-version at the grain level, only at the slice level. But it might be a nice check since grains are maintained by an external process (the "PIM").

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

Like

Reply (/forums/post?tid=5279&ReplyPostID=5312&SpaceID=127) Not answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
11/2/2019

I've edited the above post (https://autocare.communifire.com/spaces/127/sandpiper/forums/api/5279/proposal-for-slice-aggregation-terms-update?postid=5312#5312) on delivery-versioning several times as my thinking has evolved. I know this approach puts a lot of responsibility on the PIM, but as the "creator", I think it must take on that responsibility.

I've also been thinking about what Sandpiper tables might be helpful for the PIM to manage a data-pool. Chime in if I'm wrong, but I think maybe that's what **Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)** was after in this thread when adding more structure to the pool (i.e. "Slice Aggregation").

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

Like

Reply (/forums/post?tid=5279&ReplyPostID=5313&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
11/3/2019

So the problem is how to add structure to the pool so the PIM can manage the data-objects (aka "grains").

One thought is to leverage the existing (but little-used) product-group codes in the PIES Item segment (B60 & B61). This defines a two-level hierarchy for you to group parts.

B60	Product Group Code	O	AN1/10	<Group>	W12	Manufacturer-assigned Major Product Category
B61	Product Sub-Group Code	O	AN1/10	<SubGroup>	W123	Manufacturer-assigned Minor Product Category

We could then add a Sandpiper table "prodgroup_slice" to associate SubGroups to slices. This way, we don't need to assign part numbers directly to a slice.

But we might need finer control than these groupings. What if we don't sell all of a SubGroup to a trading partner, or maybe they sell it under a private label program. This means we need to assign a **subset** of a product group to a slice, and optionally give it a different identification. That's a lot for the PIM to keep track of.

One solution might be to use the existing PIES "BaseItemNumber" (B05), PartNumber (B15) and BrandAAIAID (B20) fields. For example, here we sell some of our premium Brake Pads to NAPA under their "NAPA Proformer" brand (notice we only sell them ASP1414 under this program):

Group	SubGroup	BaseItemNumber	PartNumber	BrandCode
Brake Friction	Brake Pads	ACT1414		
Brake Friction	Brake Pads	ASP1414	PF1414	GFKC

With this relationship defined, the Sandpiper Admin UI could assign SubGroups to slices, and the PIM would know which data-objects to maintain. (The data-objects in this case would be everything for PF1414).

Thoughts?

Like

Reply (/forums/post?tid=5279&ReplyPostID=5314&SpaceID=127)   Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
11/4/2019

I've been thinking a lot about this too, over the weekend, particularly your spot-on point here and elsewhere that we have to prioritize what we implement.

I still do think the structure is necessary but you are absolutely correct that the PIM will be the place this ultimately is defined. The question is more, how do we codify it and provide consistency, and when do we need to do that? For initial launch we will be more standardizing what already exists (single outputs from PIMs) and defining a new standard for interaction around "real" databases (Level 2). So perhaps the structure the way I've defined it is reaching too far forward. At the same time, I don't want to leave it unimplemented, because without at least a vision and a stub, I fear it'll never come to fruition.

What you're discussing re:deeper levels of slicing reminds me of my original thought a few months ago, to do the slice as a nested element. But maybe we can do this in a simpler way that reserves the space, leaves the option open, but doesn't specify all the system links across standards yet -- leaving that for Sandpiper 1.1. Instead we could use free-text descriptions.

I think we're good on the idea of the Pool & Set. For the Section, though, how about if that's our nested element, at this point is only a *descriptive* structure, and the slice is at the very bottom? That way it's trivial to parse with xpath: to ignore structure altogether, just parse each element from "//Slice". Each element can have a "Packed" description type that embeds the structure in a way that makes sense to the pool owner. Here's a contrived example:

```
<Sandpiper>
    <Node>
        <Pool id="1">
            <Set id="1">
                <Descriptions>
                    <Description type="Full">Farm</Description>
                    <Description type="Packed">Farm</Description>
                </Descriptions>
                <Section>
                    <Section id="1">
                        <Descriptions>
                            <Description type="Full">Cow</Description>
                            <Description type="Packed">Farm: Cow</Description>
                        </Descriptions>
                        <Section id="2">
                            <Descriptions>
                                <Description type="Full">Goes</Description>
                                <Description type="Packed">Farm: Cow: Goes</Description>
                            </Descriptions>
                            <Slice id="1">
                                <Descriptions>
                                    <Description type="Full">Moo</Description>
                                    <Description type="Packed">Farm: Cow: Goes: Moo</Desc
ription>
                                </Descriptions>
                            </Slice>
                        </Section>
                        <Section id="3">
                            <Descriptions>
                                <Description type="Full">Eats</Description>
                                <Description type="Packed">Farm: Cow: Eats</Description>
                            </Descriptions>
                            <Slice id="2">
                                <Descriptions>
                                    <Description type="Full">Grass</Description>
                                    <Description type="Packed">Farm: Cow: Eats: Grass</De
scription>
                                </Descriptions>
                            </Slice>
                            <Slice id="3">
                                <Descriptions>
                                    <Description type="Full">Hay</Description>
                                    <Description type="Packed">Farm: Cow: Eats: Hay</Desc
ription>
                                </Descriptions>
                            </Slice>
                        </Section>
                    </Section>
                </Section>
            </Set>
        </Pool>
    </Node>
</Sandpiper>
```

This reserves the space and techniques necessary to do deeper work with 1.1.

Like

Reply (/forums/post?tid=5279&ReplyPostID=5315&SpaceID=127)   Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
11/4/2019

What might help me is if we can define all the possible slices we might want to define (or at least as many examples as we can think of). For example:

- 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)

It seems to me the last one would provide the finest level of control possible, but would be static and tedious to maintain. By "static", I mean it would not dynamically include new part numbers without explicitly including them. Whereas dynamic sets would use **groupings** rather than **part number lists** to define inclusion.

EDIT: Added filter by "part-attribute".

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

1 **Unlike**

Reply (/forums/post?tid=5279&ReplyPostID=5316&SpaceID=127)   Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
11/15/2019

I've been thinking about this over the last week and decided that you're right about the scale of the effort required to build rich segmentation into the standard. I do think it still needs to be done, but not for this round.. what's more important in Sandpiper 1.0 is getting the actual transactions enabled.

I especially appreciate your thoughts on the nature of static vs. dynamic segmentation. I think you're right that it'll be good to list out the possibilities for slicing. I'll take part in that in another message.. perhaps we should start a new thread on it?

The plan is the place for static segmentation, then, which needs a lot more work before we can really implement it. Therefore I think we should reserve the structures in the plan that enable this segmentation, as stubs without the full scope of attaching metadata etc., and make sure that the communications & work done on implementation take into account that for Sandpiper 1.1 there will be more added there.

So the Section will be an element that can contain other Sections, and so on, and at this point will be kind of like <div> tags in HTML. The leaf node, the slice, is the only thing that can contain actual links to the data itself.

The exception to these divs' "stubbiness" will be that they can & must have descriptions and IDs associated with them. A program implementing Sandpiper 1.0 can simply ignore these, selecting only the slice nodes (via xpath like "//Slice" or a subsetting operation on the XML document from an object-oriented language), and the user can similarly just create a basic single Section:

```
<DataSet DataCategory="Fitment" DataType="ACES">
    <Descriptions>
        <Description Type="Full">ACES Applications</Description>
        <Description Type="Short">ACES</Description>
    </Descriptions>
    <System SystemID="Autocare">
        <SubSystem SystemID="VCdb" SystemVersion="9/27/2019" id="1" />
        <SubSystem SystemID="PCdb" SystemVersion="9/27/2019" id="2" />
        <SubSystem SystemID="Qdb" SystemVersion="9/27/2019" id="3" />
        <SubSystem SystemID="Brand Table" SystemVersion="10/1/2019" id="4" />
    </System>
    <Sections>
        <Section id="1">
            <Descriptions>
                <Description Type="Full">Brake Products</Description>
                <Description Type="Short">Brake</Description>
            </Descriptions>
            <Slice uri="https://sandpiper.bapbrakes.com/AABBCCDD/AA" id="1">
                <Descriptions>
                    <Description Type="Extended">Bob's Auto Parts: ACES Applications: Bra
ke Pads &amp; Shoes</Description>
                    <Description Type="Full">ULTRA Braking Solutions</Description>
                    <Description Type="Short">ULTRA</Description>
                </Descriptions>
                <Links>
                    <LinkEntry SystemID="4" KeyFieldValue="XXXB">Autocare Brand: ULTRA Br
ake Premium</LinkEntry>
                    <LinkEntry SystemID="4" KeyFieldValue="XXXC">Autocare Brand: ULTRA Br
ake Value</LinkEntry>
                    <LinkEntry SystemID="2" KeyFieldValue="1234">Autocare PCdb: Disc Brak
e Pad</LinkEntry>
                    <LinkEntry SystemID="2" KeyFieldValue="5678">Autocare PCdb: Disc Brak
e Pad Set</LinkEntry>
                    <LinkEntry SystemID="2" KeyFieldValue="1688">Autocare PCdb: Drum Brak
e Shoe</LinkEntry>
                </Links>
            </Slice>
            <Slice uri="https://sandpiper.bapbrakes.com/AABBCCDD/BB" id="2">
                <Descriptions>
                    <Description Type="Extended">Bob's Auto Parts: ACES Applications: Bra
ke Pads &amp; Shoes: PERFORM</Description>
                    <Description Type="Full">PERFORM Braking Solutions</Description>
                    <Description Type="Short">PERFORM</Description>
                </Descriptions>
                <Links>
                    <LinkEntry SystemID="4" KeyFieldValue="XXXD">Autocare Brand: PERFORM
 Brake</LinkEntry>
                    <LinkEntry SystemID="2" KeyFieldValue="1234">Autocare PCdb: Disc Brak
e Pad</LinkEntry>
                    <LinkEntry SystemID="2" KeyFieldValue="5678">Autocare PCdb: Disc Brak
```

```
e Pad Set</LinkEntry>
                    <LinkEntry SystemID="2" KeyFieldValue="1688">Autocare PCdb: Drum Brak
e Shoe</LinkEntry>
                </Links>
            </Slice>
            <Slice uri="https://sandpiper.bapbrakes.com/AABBCCDD/BB" id="3">
                <Descriptions>
                    <Description Type="Extended">Bob's Auto Parts: ACES Applications: Bra
ke Rotors &amp; Drums</Description>
                    <Description Type="Full">Brake Rotors &amp; Brake Drums</Description
>
                    <Description Type="Short">Brake Iron</Description>
                </Descriptions>
                <Links>
                    <LinkEntry SystemID="2" KeyFieldValue="1744">Autocare PCdb: Brake Dru
m</LinkEntry>
                    <LinkEntry SystemID="2" KeyFieldValue="1896">Autocare PCdb: Disc Brak
e Rotor</LinkEntry>
                    <LinkEntry SystemID="2" KeyFieldValue="10292">Autocare PCdb: Disc Bra
ke Rotor and Hub Assembly</LinkEntry>
                </Links>
            </Slice>
        </Section>
    </Sections>
</DataSet>
```

Like

Reply (/forums/post?tid=5279&ReplyPostID=5322&SpaceID=127)   Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
11/15/2019

Another note, we could have the main client for 1.0 simply automatically create one section, that has the same description as the pool or something similar. And not expose this functionality at all in our early work here.. just document it in the schema and make sure that developers know to not hardcode an expectation for just one level of segments.

Like

Reply (/forums/post?tid=5279&ReplyPostID=5323&SpaceID=127)   Answer