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

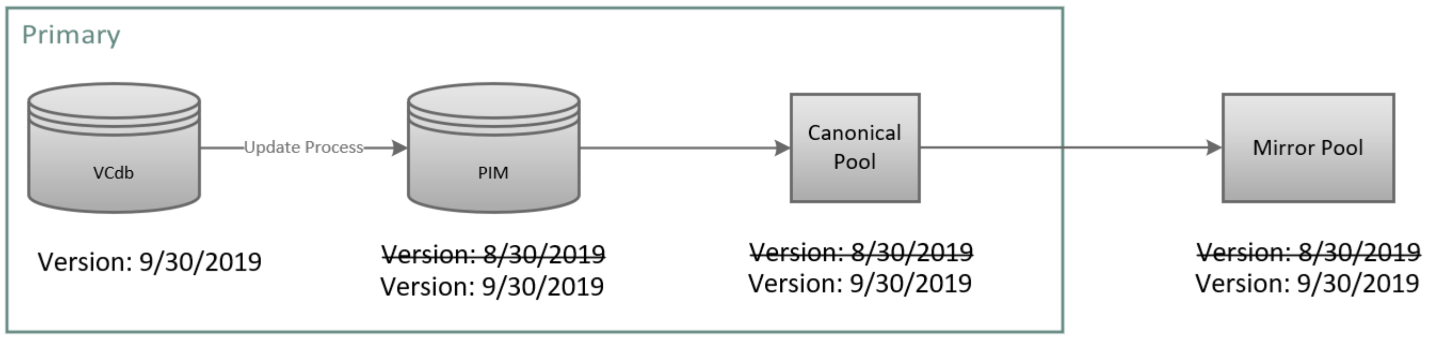
Referencing Standard DBs and Versioning

Unsubscribe from this discussion
Subscribe to RSS (../../../../../../spaces-cf/forums/rss-space-posts.ashx?spaceID=127&topicID=5223&forumID=5044&key=rrer%2B1xDo%2BlxpC7jBRbM5w%3D%3D)

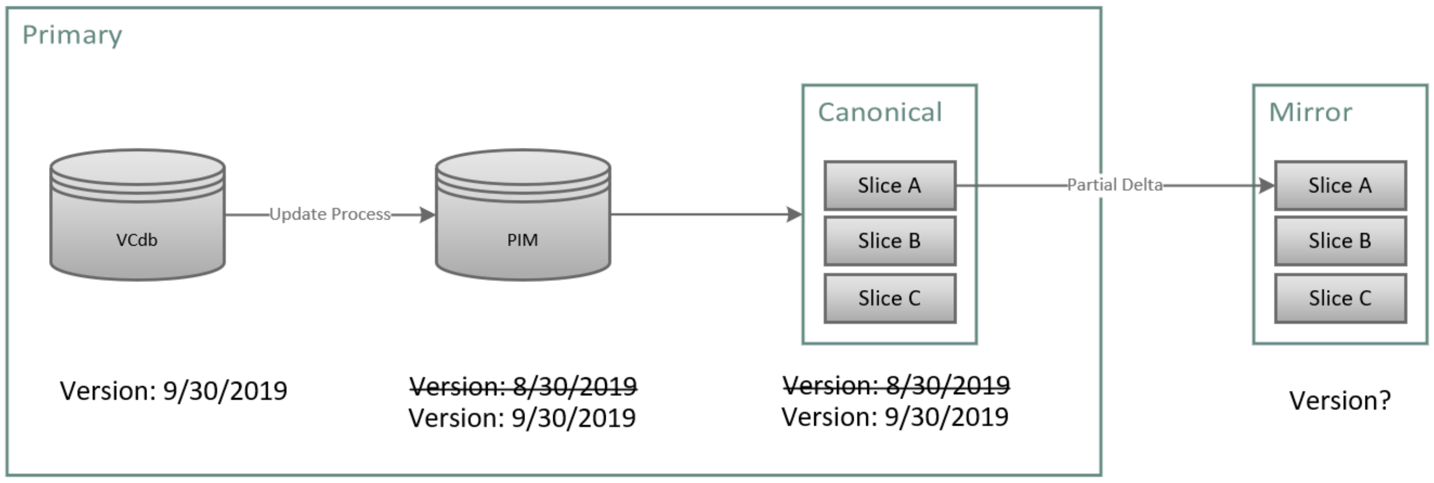
Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
10/2/2019

On our call today we talked about the difficulty of ensuring that referenced data remains valid even as its versions change, and possibly change the meaning or existence of the reference.

This isn't so much a problem when replacing the whole pool:



But it becomes an issue when doing small updates to just one slice. How do you determine the version to reference if you do this?



Like

Reply (/forums/post?tid=5223&ReplyPostID=5224&SpaceID=127)

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
10/2/2019

My thoughts on this: The version itself is a change that affects all data, because you've gone through an update process that has examined all the data in the canonical pool and removed or altered it as required.

The slices should each get the new version applied. The only question is, how best to do this?

I talked about sending an add/delete on the version for every slice, but this would have no GUID so by our definition it couldn't be part of the existing set-based process. But then we'll need some way to communicate this change, and maybe it doesn't belong as part of a change set anyways. Maybe there's a special kind of data that we can include in an API operation, guaranteed unique fields -- kind of like the header in the ACES and PIES files, it could be in the metadata about the slice itself. And an update to that metadata could trigger a synchronization.

Dave raised the idea of having the version stored per-record, and how that seems like a solution but it's far outside what an e-tailer would have implemented, and adds a lot of overhead. In my own work I've taken a different approach, where I have snapshots of our data and the specific valid database versions it points to. I'm not sure that works here..

Like

Reply (/forums/post?tid=5223&ReplyPostID=5225&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
10/4/2019

Edit: Head over to this post (https://autocare.communifire.com/spaces/127/sandpiper/forums/api/5242/pools-vs-slices) to discuss elements of the pool vs. the slice. Leaving this here so there's context, but the discussion about parts outside the reference data portion continues elsewhere

---

In thinking more, this has helped me to change how I was thinking about the pool vs. a slice. It's given me a good way to approach the XML manifest as a way to solve some of this. Here's a proposal.

First, a pool should be one type of data. A single primary node should have multiple canonical pools, one for each type -- for example, one ACES application pool, one NAPA application pool, one PIES pool, etc. -- because the categorization, processing, and storage requirements for each of these will be vastly different.

The pool itself then explicitly states the system and subsystem that it references, as well as their versions. Autocare would be the system, and the subsystem is PCdb, VCdb, PAdb, Qdb, Branding, etc. The versions are their dates in this case; other systems would use other version schemes potentially. (How to handle schema versions -- I think in the same structure. Need to decide that)

The pool is actually a special kind of slice; a slice of all the data types, that is required to make that one special data type declaration.

Each pool then contains one or more slices. A slice can reference key values from the systems defined at the pool level, but at each level, only one *type* of reference can be made. For example, all of the first level slices might reference brand owner from the Autocare branding table, but they may not reference any other field, to ensure that they are a true A-B choice using the same kind of decision. There is a nice way to do this in the XML.

For an imaginary supplier Bob's Auto parts that wants to segment their data this way (note I'm only going down to detail in one of each slice below, presumably every one could go deeper):

1. Bob's Auto Parts ACES Applications
    1. Bob's Brakes
        1. ULTRA Brakes

2. PERFORM Brakes
    1. Brake Friction
    2. Rotors
    3. Drums

The XML could define this:

```
   <Pool DataCategory="Fitment" DataType="ACES">
      <Descriptions>
         <Description Type="Full">Bob's Auto Parts: ACES Applications </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>
      <Slices>
         <SystemLink LocalSystemID="4">Autocare Brand Table</SystemLink>
         <Slice uri="https://sandpiper.bapbrakes.com/AABBCCDD">
            <Descriptions>
               <Description Type="Full">Bob's Auto Parts: Brakes</Description>
               <Description Type="Short">BAP Brakes</Description>
            </Descriptions>
            <Links>
               <LinkEntry LocalSystemID="4" KeyFieldID="BrandOwner" KeyFieldValue="XXXA">Bob's
Brakes</LinkEntry>
            </Links>
            <Slices>
               <SystemLink LocalSystemID="4" KeyFieldID="Brand"/>
               <Slice uri="https://sandpiper.bapbrakes.com/AABBCCDD/AA">
                  <Descriptions>
                     <Description Type="Full">ULTRA Braking Solutions</Description>
                     <Description Type="Short">ULTRA</Description>
                  </Descriptions>
                  <Links>
                     <LinkEntry KeyFieldValue="XXXB">ULTRA Brake Premium</LinkEntry>
                     <LinkEntry KeyFieldValue="XXXC">ULTRA Brake Value</LinkEntry>
                  </Links>
               </Slice>
               <Slice uri="https://sandpiper.bapbrakes.com/AABBCCDD/BB">
                  <Descriptions>
                     <Description Type="Full">PERFORM Braking Solutions</Description>
                     <Description Type="Short">PERFORM</Description>
                  </Descriptions>
                  <Links>
                     <LinkEntry KeyFieldValue="XXXD">PERFORM Brake</LinkEntry>
                  </Links>
                  <Slices>
                     <SystemLink LocalSystemID="2" KeyFieldID="PartTerminologyID"/>
                     <Slice uri="https://sandpiper.bapbrakes.com/AABBCCDD/BB/AA">
                        <Descriptions>
                           <Description Type="Full">Bob's PERFORM Brake Friction</Description>
                           <Description Type="Short">PERF Brake Friction</Description>
```

```
                </Descriptions>
                <Links>
                  <LinkEntry KeyFieldValue="1234">Disc Brake Pad</LinkEntry>
                  <LinkEntry KeyFieldValue="5678">Disc Brake Pad Set</LinkEntry>
                  <LinkEntry KeyFieldValue="91011">Electronic Wear Sensor</LinkEntry>
                </Links>
              </Slice>
              <Slice uri="https://sandpiper.bapbrakes.com/AABBCCDD/BB/BB">
                <Descriptions>
                  <Description Type="Full">Bob's PERFORM Brake Rotors</Description>
                  <Description Type="Short">PERF Brake Rotors</Description>
                </Descriptions>
                <Links>
                  <LinkEntry KeyFieldValue="121314">Disc Brake Rotor</LinkEntry>
                  <LinkEntry KeyFieldValue="151617">Disc Brake Rotor and Hub
Assembly</LinkEntry>
                </Links>
              </Slice>
              <Slice uri="https://sandpiper.bapbrakes.com/AABBCCDD/BB/CC">
                <Descriptions>
                  <Description Type="Full">Bob's PERFORM Brake Drums</Description>
                  <Description Type="Short">PERF Brake Drums</Description>
                </Descriptions>
                <Links>
                  <LinkEntry KeyFieldValue="181920">Brake Drum</LinkEntry>
                </Links>
              </Slice>
            </Slices>
          </Slice>
        </Slices>
      </Slice>
    </Slices>
  </Pool>
```

Like

Reply (/forums/post?tid=5223&ReplyPostID=5234&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
10/8/2019

> "Maybe there's a special kind of data that we can include in an API operation, guaranteed unique fields -- kind of like the header in the ACES and PIES files, it could be in the metadata about the slice itself." **Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)**

I believe this is an important part of the solution to the "reference file" problem. Each sync request should return metadata that includes any dependencies.

Another piece of the solution, I think, is ensuring that the entire pool uses the same reference file version. Otherwise, it might be possible to change a Slice definition and have it become inconsistent (some objects using one reference file version and others using a different file version).

consistency (/spaces/127/sandpiper/searchresults?keyword=%22consistency%22&searchtags=1)
metadata (/spaces/127/sandpiper/searchresults?keyword=%22metadata%22&searchtags=1)
sync (/spaces/127/sandpiper/searchresults?keyword=%22sync%22&searchtags=1)

Like

Reply (/forums/post?tid=5223&ReplyPostID=5240&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
10/9/2019

Remember that any data operation at any level should still happen in the context of the slice. So as long as that slice's references are stated, each operation's references are carried with.

I could see a need for the pool to carry different versions of a database, if there are some sections of data that have not been updated and will not be updated at the same pace. I'm thinking in particular of those that are frozen because a given partner wants them to stay on an older version or slower speed.. and this would become more and more important as we discuss the idea of more-frequent Autocare db updates. Some folks will stay on a slower cadence and require their transmissions to act the same way.

So one of these slow consumers might get a freeze-dried version of the main line, with only critical or reference-independent fixes being made (like obsolete parts being removed, mfrlabel updates, etc)

Like

Reply (/forums/post?tid=5223&ReplyPostID=5246&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
10/9/2019

✓ Answered

> "I could see a need for the pool to carry different versions of a database, if there are some sections of data that have not been updated and will not be updated at the same pace"
> **Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)**

That does complicate matters.

One way to handle it is to not allow a data-object to be shared between slices. So each data-object would be assigned to a single Slice, and that Slice would define any dependencies (reference data versioning).

If the Slice definition changes (e.g. a part-category is added) new data-objects are created and added to the Slice.

This does put more work on the PIM, however, to add and drop these data-objects wherever needed.

Does anyone see a better solution?

1 **Unlike**

Reply (/forums/post?tid=5223&ReplyPostID=5247&SpaceID=127) Not answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
10/9/2019

> **On 10/9/2019 11:46 AM, Doug Winsby said:**
>
> If the Slice definition changes (e.g. a part-category is added) new data-objects are created and added to the Slice.

I think there are two kinds of changes to the slice: altering the hierarchy and its fundamental nature, and altering the links it has to outside systems.

The former creates a new slice, essentially; the fundamental nature of the slice should be immutable. Adding or removing subslices, adding or removing siblings, etc. -- these are changes that require the plan to be re-communicated and agreed to by both parties, as you're changing the foundation.

In contrast, system links simply ties the slice to other standards, and doesn't change what the slice actually *is*. It's just adapting to other versions of the standards, or new standards. The secondary actor sometimes needs to be acutely aware of this if their system depends on those links, but the slice itself is still the same within Sandpiper. So in that regard the slice object remains the same even if the links change.

For example, if I'm getting brake calipers and rotors from a manufacturer, and they have these sub-sliced into Calipers vs. Rotors, it doesn't matter if the manufacturer links to a new Autocare "Caliper and Bracket Assembly" part type; the caliper category already included that kind of product, it just wasn't available to link to.

In this same way, after typing this all out, I'm leaning toward the version of those standards actually occurring somewhere higher than the slice itself; that gets back to our pool vs. slice discussion, so I'll leave the point here and we can pick it up there..

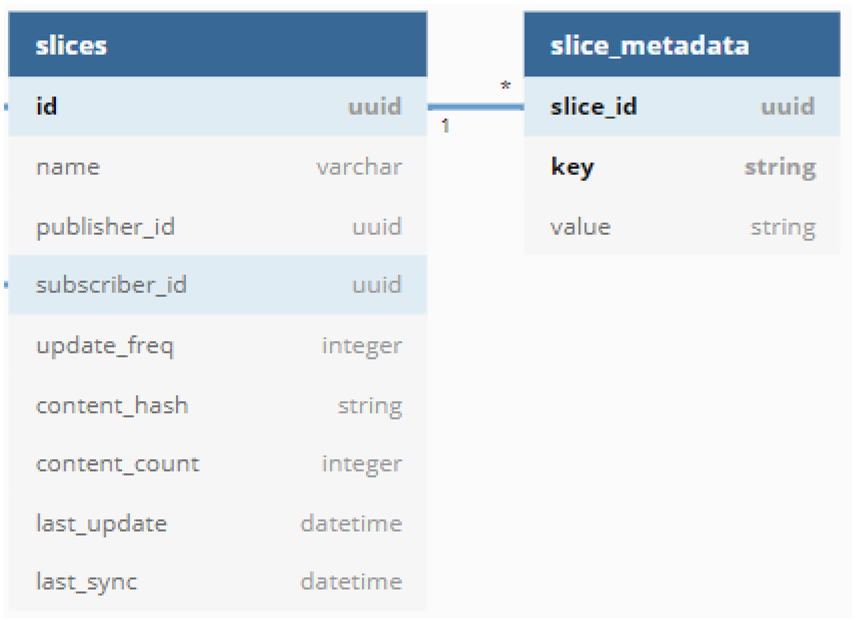
Like

Reply (/forums/post?tid=5223&ReplyPostID=5248&SpaceID=127)   Answer

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

I think we all agree that we need the ability to tie "reference versions" to our data-objects (or their containers) at some level.

I like the idea of using key-value "meta-data" to handle this requirement. Here I've shown how that might look if the level was a "Slice":



slices	
id	uuid
name	varchar
publisher_id	uuid
subscriber_id	uuid
update_freq	integer
content_hash	string
content_count	integer
last_update	datetime
last_sync	datetime

slice_metadata	
slice_id	uuid
key	string
value	string

With this structure, we could have, for example:

```
{slice_id: "cc27830b-f9ec-4cff-a11f-a5cda4a5bddf", key: "vcdb-version", value: "20
19-09-27"}
{slice_id: "cc27830b-f9ec-4cff-a11f-a5cda4a5bddf", key: "pcdb-version", value: "20
19-09-27"}
```

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

Like

Reply (/forums/post?tid=5223&ReplyPostID=5256&SpaceID=127)   Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/18/2019

Asking Sandpiper for your subscribed slices might return something like the following:

```
// http://sandpiper.example.com/v1/slices (http://sandpiper.example.com/v1/slices)

{
  "slices": [
  {
    "slice-id": "08efdf90-a815-4cf7-b71c-008e5fd31cce",
    "slice-name": "AAP-Brakes",
    "slice-hash": "cf23df2207d99a74fbe169e3eba035e633b65d94",
    "metadata": {
      "pcdb-version": "2019-09-27",
      "vcdb-version": "2019-09-27"
     },
     "count": 2919
   },
  {
    "slice-id": "cb4b768b-6d6b-4965-a29a-9052a80dbbbb",
    "slice-name": "AAP-Wipers",
    "slice-hash": "1a804c61e1a70ab37b912792ee846de7378c4a36",
    "metadata": {
      "pcdb-version": "2019-09-27",
      "vcdb-version": "2019-09-27"
     },
     "count": 2342
   }
  ],
  "count": 2
}
```

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

Like

Reply (/forums/post?tid=5223&ReplyPostID=5403&SpaceID=127)   Answer