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

The Sync Process

Unsubscribe from this discussion
Subscribe to RSS (../../../../../../spaces-cf/forums/rss-space-posts.ashx?spaceID=127&topicID=5146&forumID=5044&key=rrer%2B1xDo%2BlxpC7jBRbM5w%3D%3D)

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
8/27/2019

I previously posted this pseudo code for a sync:

```
authenticate   // this is a topic for another discussion tread
get my.slices
for each slice in my.slices
  get slice.hash    // sha1 or md5 hash representing slice oids
  if slice.hash <> local.slice.hash    // saved from last sync or must we recalc?
    get slice.oid_list
    compare slice.oid_list with local.slice.oid_list
    for each new_oid in slice.oid_list
      get object.new_oid     // this object contains the payload
      store object in local.slice
    remove all obsolete local.slice.objects
  put slice.sync_completed   // for activity reporting purposes
```

Luke was concerned that the inner loop (which retrieves a single object) would require too many rapid-fire calls and so suggested we batch that request (in a POST with oids in the body).

In the DataObject discussion, I gave an example of what might be returned by a batch request of objects:

```
{
     "header": {"slice-id":"BPI-WHI", "index": 1, "count": 5000},
     "objects": [
         {
             "oid": "GQBFRvf112p33Q", "type": "PartsProApp", "created": "2019-08-1
2T11:30:00Z",
             "keys": [ "P1234" ],
             "payload": "MQkxOTk1CTEJNQkJCQkJCQkJCQkJCQk3NDE4MglNR0MJNDU3M ..."
         },
         {
             "oid": "GQBFRvf112px83a", "type": "PartsProApp", "created": "2019-08-
14T09:30:10Z",
             "keys": [ "P4321" ],
             "payload": "MQkxOTk1CTEJNQkJCQkJCQkJCQkJCQk3NDE4MglNR0MJNDU3M ..."
         }
     ]
 }
```

One problem this presents, however, is how to indicate a bad request. What happens if one or more of the oids requested do not exist? I understand this should never happen (because we just got a list of items to sync, why would we ask for something not in that list), but it is something we must plan for.

Maybe we could include an array of "notfound" ids. But should we return that in each of the returned batches (assuming we need to page the results). These are the little decisions that make designing an API so much fun!

Like

Reply (/forums/post?tid=5146&ReplyPostID=5147&SpaceID=127)

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
8/28/2019

> **On 8/27/2019 2:59 PM, Doug Winsby said:**
>
> One problem this presents, however, is how to indicate a bad request. What happens if one or more of the oids requested do not exist? I understand this should never happen (because we just got a list of items to sync, why would we ask for something not in that list), but it is something we must plan for.

I'd say it almost should cause a termination of the differential sync process and initiate a full refresh, or an override by a human. It indicates a failure outside the realm of recovery.. something sinister lurking.

Like

Reply (/forums/post?tid=5146&ReplyPostID=5148&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
8/28/2019

> ***On 8/28/2019 8:36 AM, Krister Kittelson said:***
>
> "I'd say it almost should cause a termination of the differential sync process and initiate a full refresh..."

That may be too harsh. Requesting an obsolete object-id could easily happen if the sync was performed during a PIM update of the data-pool. I do like the idea of "terminating the sync" and starting over, though. A "full refresh" should really only be needed when a secondary-data-pool is initiated, corrupted or lost.

I think an http response code 410 works in this situation:

> **410 Gone.** Indicates that the resource requested is no longer available and will not be available again. This should be used when a resource has been intentionally removed and the resource should be purged. Upon receiving a 410 status code, the client should not request the resource in the future.

This would indicate to the client that one of the object-ids was bad and they should request a new list of ids. The server should also log these errors (in case there is a bug in the server logic).

Like

Reply (/forums/post?tid=5146&ReplyPostID=5150&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
8/28/2019

This is how others have solved the batch issue:

- Google Drive (https://developers.google.com/drive/web/batch) (note: only mixed requests are deprecated)
- Facebook (https://developers.facebook.com/docs/graph-api/making-multiple-requests)
- Microsoft (https://msdn.microsoft.com/en-us/library/office/dn903506.aspx)

It looks like all of them return a http result code for each item in the batch. This might be a good pattern to follow.

Like

Reply (/forums/post?tid=5146&ReplyPostID=5152&SpaceID=127)   Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
11/30/2019

My current thinking for sync processing is to use web-sockets (https://blog.teamtreehouse.com/an-introduction-to-websockets). A traditional HTTP RESTful (stateless) approach really doesn't fit the requirements. It would include a lot of unnecessary overhead for each request (e.g. dns resolution, a new connection, ssl handshake, etc.). This is not required when you have two well-known processes from pre-determined end-points trying to communicate.

> "The client establishes a WebSocket connection through a process known as the WebSocket handshake. This process starts with the client sending a regular HTTP request to the server. An `Upgrade` header is included in this request that informs the server that the client wishes to establish a WebSocket connection."
>
> "[Once] the handshake is complete the initial HTTP connection is replaced by a WebSocket connection that uses the same underlying TCP/IP connection. At this point either party can start sending data."

Easy-peasy! (We'd still use token-based authentication and "wss", which uses https for security).

Using RPC (remote procedure calls) over web-sockets is one way to define callable API "end-points" (similar to REST resources). Furthermore, we'll need to define the message package format. Web sockets supports binary data, so we have lots of options here. Serialization technologies we're considering are JSON-RPC (https://www.jsonrpc.org/), BSON (http://bsonspec.org/), Protocal Buffers (https://developers.google.com/protocol-buffers/) and MessagePack (https://msgpack.org/).

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

Like

Reply (/forums/post?tid=5146&ReplyPostID=5365&SpaceID=127) Answer

Luke Smith (https://autocare.communifire.com/people/autopartsource)
12/8/2019

JSON-RPC looks like a sane choice. My opinion is that anything needing full 8bit "binary" representation (actual asset content?) should be done in base64.

Like

Reply (/forums/post?tid=5146&ReplyPostID=5382&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/8/2019

The serialization portion of RPC is really a complicated topic. The direction seems to be toward external schema-driven formats (like protobuf (https://developers.google.com/protocol-buffers/), thrift (https://thrift.apache.org/) and webrpc (https://github.com/webrpc/webrpc/blob/master/schema/README.md)). I like the "contract" concept these offer, but they usually use code generation (which adds extra tooling and build complexity). They're also *usually* a binary format over the wire which can make debugging more difficult.

JSON is a good choice because it has a lot of support and is human-readable. It can be slower to encode/decode to internal structures, though.

MessagePack might be a good compromise, but whatever we choose, it would be simple to replace.

This is probably a case where we should use the "Don't optimize prematurely" design pattern. If we have a bottleneck with JSON, we can look at alternatives.

I agree that `base64` seems like a good choice for encoding binary data if using JSON over the wire.

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

Like

Reply (/forums/post?tid=5146&ReplyPostID=5383&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
12/9/2019

JSON does seem like it scales OK and is pretty well supported. If we come down to it maybe 2.0 could support a binary "high-perf" version but especially for debugging and development the JSON plain-text would make life a lot easier, I imagine.

Like

Reply (/forums/post?tid=5146&ReplyPostID=5385&SpaceID=127) Answer

Page 1 of 1 (8 items)