2019-07-02 Specification meeting

Date

24 Jun 2019

https://duraspace.zoom.us/j/593874187

Find your local number: https://zoom.us/u/X8NrH

Attendees

- Michael Ritter
- Bill Branan
- tamsin johnson
- Sibyl Schaefer

Discussion items

I

Time	Item	Who	Notes

	Notes		Issue of pull. Either model works, matter of preference. Should the relationship between bridge and DDP be pull oriented? Advantage being that it limits the interface footprint for the DDP. API demands that the Bridge makes on the DDP are reduced. Thoughts: API expectations are reduced, but implementation effort is not reduced because it has to implement whatever the pull mechanism is. Risk of hidden expectations wrt how the Bridge and the DDP determine that something has been preserved. Needs to be a common understanding of state across all systems. This exists whether the push or pull model is used. Otherwise pull is a pleasant pattern. Same question with the Gateway and the Bridge. Benefits are less compelling there wrt the API. If I'm the Bridge and pulling the manifest I want to get a listing of objects that I'm going to want to preserve so I can go get those individually, want to continually pull for that manifest. When I see an object again, I can look at that as a request to pull it again as a version. Latency issues. Advantages of push - presents a lot more opportunity for the Bridge to respond. If Gateway initiates request to preserve, http response gives indication if it's not a good request or if something is wrong. Partial pull - Gateway makes initial call, here's the set. Then Bridge pulls individual objects. If transfer fails, Bridge can try again. Decision -> move forward with the push manifest approach. Between Gateway and the Bridge. Same argument apply to the Bridge and DDP? Model that is in place right now - do pull based on the state of the deposit. Question is either the DDP would have to track the deposits its working on, or mechanism in Bridge to change state. DDP queries for things in that state. Worth re-visiting some type of pub/sub? Bridge can push notification out, DDP can receive and work on it. Nice thing about the current model, no need to worry about a missed notification. Is possible for messages to get dropped with pub/sub. Current API allows us to push up history events. Can gather where we were in the flow if failure happens. Is that model valid in this case also? Where there is no API at the DDP but all the interaction happens at the Bridge API. Not having an endpoint is an attractive prospect. Now: **waiting for DDP** state value. API on bridge to query for all active snapshots. Query on state, all those are things that need to be deposited. In addition, API call to get history items on each snapshot. Use to check what place we are in the processing. DDP gets listing of objects that are waiting. Once request is received, iterate each object that needs to be preserved and get additional info to see what state it's actually in. Additional API. Remain in Waiting for DDP until the DDP makes another call to the Bridge to say this is done. At that point, clears cache, sends notification back to Gateway. What else would we expect the DDP to communicate to the Bridge about? Audit events. How much should we be pushing to audit events? As long as we have the mechanism for introducing audit events, detail of what those events are require more digging to know what those events are. Frequency, etc. What are the needs at the repository layer with asking for that info. Audit endpoint - assumption that it will have whatever synchronous data is available. Mechanism in mind was to store it essentially as an access cache in the Gateway. Gateway receives whatever info is needed to make synchronous requests accessible in quasi-realtime. Canonical source of audit info is the DDP or DDP via the Bridge. Gateway services repos immediate needs. Assumption is info from the DDP is slow. Can we get away with not having an API for the DDP? What would be necessary that we couldn't already get from the OtM API as its listed now. Need to add a way to update the deposit - have the final notification that the deposit has been preserved. Or you need the DDP to be sending audit events that allow the Bridge to update that status independently. REST call as opposed to an audit event - gets back an error and retries rather than pumping out events which might get missed. Calls that are added in with question marks - there's a certain set of calls that flow as part of the deposit- abort, restart. Do we need those same calls for deletes and restores? Use case for abort - if deposit is started and then fails. Have an option to either restart or abort which means stop trying to process this thing. Notification needed. Poll deposit for its status? Get audit info delivered in someway or the other? Abort means you can delete files. Retry means retry the file that failed, not the one that didn't. Need some sort of timeout. Avoid resending solved by Bridge knowing the checksums of the objects that it has in its cache. Right now each post deposit gets back a deposit ID. If you have the same file ID in two deposit IDs, you wouldn't want to pull the file twice. What's the result of that? Even outside of failure. Bridge has to keep track of a single file being in multiple deposits. Completion state- one before the other, can't delete the one file. Implementation detail. Design goal for gateway - that it doesn't need to know anything about deposit states. Trying to manage the state of deposits means that state bleeds across the two systems. Not sure it's easy to achieve. Failure that needs manual intervention - what is needed to get started again? Gateway - has a bunch of objects that it needs to make sure are deposited. Go to Gateway with push deposits some content available as cache. Failure during deposit, Gateway is notified or polled. Proposal - partially complete deposits with errors remain in that state forever.
	Bill's summary		We made 2 decisions: • Stick with the interaction model that has the Gateway POST a request to the Bridge and the Bridge pull content • Don't plan to create an additional DDP API. Instead allow all of the interactions to flow through the Bridge API. We left with 2 interrelated points for continued discussion: • Should we expect the Gateway to be aware of state in the Bridge such that it handles actions like restarting or aborting deposits or instead allow the Gateway to remain as free as possible from knowledge of those states? • Should we allow parts of deposits to succeed independent of other parts?
	Deliverables for July 22 meeting		(1) The Bridge and Gateway API docs reasonably complete (2) An executive summary of how the entire system will function (3) A walk-through of the interaction flow from repository down to DDP, including which API calls would be used, for each of the primary flows (deposit, restore, delete). This should include both narrative and diagrams.

Action items

-