openIDL - Architecture Discovery (Drawing)

Discussion notes:

AWG Master diagram:

https://lucid.app/lucidchart/ac20d4e1-50ad-4367-b5cf-247ed9bad667/edit?viewport_loc=-301%2C-37%2C3200%2C1833%2CkgLSxXlpoGmM&invitationId=inv_de1a5e61-8edc-488a-90ee-8312f8c69cd4#

- KS -
    - Work from this and other diagrams
    - Draw boxes as opposed to blank slate
- PA - start broad, draw stuff until we have off
- KS - can check off major functions, components in swim lanes, need to ingest fata, load it, data calls, extract, report - high level - can start from left and spitball flow of data as it comes through the system, get high level boxes in there, or as we go itemize whats in boxes - if doing it alone Ken would do all the above - start from somethign may or may not be close
- KS - first step: getting data into HDS, through some means, means many mult DBs or systems are the source of this info, needs to be normalized - ETL is going to normalize data from diff sources into an openIDL format (HDS - first thing to happen) - another thing that is going to happen, we are goign to edit it (syntax, data errors in ETL) then convert to HDS format and load into HDS format (all happens in Member Enterprise on : LEFT of diagram
- JM - first point of concern, to do the edits, need to be in standardized format (for predictability)
- KS - if standardizing edits, therefore the data input must be standardized at this point, putting data into standardized format - edit package will edit it for validity aganzt a number of rules, know we have a rules engine in the ETL (and rules repository lies in ETL)
- JB - puts data into sep format and second step where it checks it
- KS - ETL standardizes data, then ETL Edit engine (Rules Engine, Rules Repo) - 2 major steps going on inside here: ETL and Edit then converter to map it to format for HDS
- KS - standardized, edited (rules engine and repo) then mapped to HDS format - coming out of here: only the valid
- JB - warnings and exceptions based on issues
- JM also intermediary data set, want to land data into some structure visible to run edit rules, some type of repo, biased towards persistence
- JB - batch or message format
- JM - stuff on fly needs troubleshooting, persisting diff issue (maintenance)
- JB - just some kind of file format with w/ schema, well-defined structure
- KS - after edit, thought we were keepoing all the records in the HDS, still true? Flagging those with errors? Or just daying
- JB - 2 types: data errors - sanity check?
- KS - ui for controlling edits and release of this stuff
- JB - going to be issues with data vut if thresholds not above a cert threshold... path, data quality checks - discussed everyhting submitted goes to HDS b/c it comes from the source
- KS - SDMA func "this has x erroers but in a few months will release all data
- JB - submissions - daily or weekly
- JM then dont need UI - up to carrier to figure out how to pass all tests,
- KS - think we heard wd rather have standardized process, need UI to get it to work - no consensus, thinks Dale
- JB could be process that reads files and provides outputs
- KS - 10k+ rules that are needed, people want to leverage
- PA - as someone who used SDMA a lot, lot to be said for how that has allowed users to self-service
- JB - supply somehing for all, add own rules, but also a standard set of rules all can use
- JM - depends on what you want in interface, put UI that allows to see whats going on is fine vs UI for editing data
- PA - doing it today, have both options,can update rows but a lot of times errors are when carrier ETL fails to make it correctly
- KS - not happenign quick enough for cycle to finish - have to get it out in the week, can't get it fixed at the source but need tp be able to hit endpoint - then DB got complicated, error log records, etc.
- JB - sep files from corrected records, if there are tests for sanity check, level of quality, up to carrier to try to resolve?
- KS - heard doesn't work, what we want to to but can't always do that - sometimes timeframe needs fix in SDMA, either data or process wont change in time
- JB - spot in the middle, can't change box 1 in pic, you can change downstream, doesn't HAVE to mean HDS, basically a box 2 instead of UI for finxing it
- KS - do it after standardization, not on carrier, up to openIDL footprint to make changes, IS IT PART of openIDL footprint to provide fixing for this data - how do we decide
- PA - flip to PATL page, kind of boiled down version, SDMA today, what we can see being robust way of doing it - Carrier ingestion portal, do large edits before or stnadardarzed edit adfter, through package run against any data set, from ingeastion portal trivers a job to HDS - as soon as we put in working table, assign UUID,
- KS - not clear where error editing is happening
- JM - how od you make changes? edits?
- KS - have UI to make changes, shows tables, almost like excel editing
- JB - who does that at carrier?
- PA - Susan or Reggie for ex at TRV - the business people in charge of loading data, something like this a feature a lot of companies would want to run, like TRV bypass whole system, smaller companies would want it
- KS - TRV currently fixing data with SDMA - every now and then we can't get the back end to feed the right data
- PA - thinks way we are right now, making all work with this workflow, TRV isn't making changes to data lake, submitting excel with why #s (adjustment artifact) why adjust should happen - as of today AAIS is not updating records in data lake, allowing to edit at load, working in ingestion portal
- KS - need to decide if thats a req for the system or not, can say "if we have it, where would it be?"
- PA - whole secondary thing in HDS, error table and correction table in HDS, not there today
- JB - baseline data qual check, simply check data for acceptibiloty (errors under threshold) and pass along if not exceeded OR if exceeded carrier would need to update, have means to edit records and proecc using tools from openIDL - baseline data qual check where data meets cert qual, if not accepted would carrier allow to be edited or back on carrier
- JM - day, 1, day 2, day 3 - simplest design, carrier loads HDS and then done - maybe set flag "ready to go" and if it fails test back up batch and put it back in, no staging area or UI needed to vchange
- JB - loading into DB and back it out, squirrely, having format to check on way in (instead of loading garbage)

- JM - babystep: v1 is flat load and back out (otherwise staging area to run rules against it) v2 fail and fix, v3 do whatever it takes (typically have aspot where in flow can modify data, OR formal facility in there - day 1, 2, 3 question) - mixing delivery with Arch
- JB - likes V2, simplest way to start, door open, if we think a modification interface, qual checs vs modifications, cant let crappy data- load, iterate across, - how do fixes re-apply if you reload stage, fixes in robust design, put them in a fixit body of tables, re-apply fixits - fixit gets big fast
- PA - non robust vers TODAY of what JM described, once you see issue, so we make the right call to reload or...
- JM - load 1x, modify with fixit interface
- JB - copy, fix it, resubmit - still have sep files
- JM - prob with fixit, copy made must have structure, could be overwritten with reload
- PA - haven't ironed out - 4 csv in one day for auto (1 month worth of loading) - are those 4 CSV one job? mult docs per job?
- JB - org'd as sep submissions or files
- KS - some scope of identifier of this package of data, work on package identifier, dont release until edit/removed
- PA - pivor slightly, passing 47 of 48 states, is there a facility fto load 47 who passed or all-or-nothing?
- JM - batch ID mech on all tables, put it into stage, its a batch - if you chose to say "loaded 48 states as a batch and one is wrong, back them all out, individually? <SEAN REVIEW TIME 45min)
- JB - file in a folder means needs to be processed
- KS - wouldn't want huge company like Hartford wouldn'et want to equate file to a batch (mult batch to file, mult file to batch)
- PA - why would a batch have a state/line indiviation - whether pass or fail 4 sep batches
- JB rules by state?
- PA - depends, rules are more national for most part but passing "is this data for this state/line/timeframe valid?
- KS - can batch be orthogonal to reporting? Sent bunch of data, want to fix it all if it doesn't work only found one state screwed up - can't pass the state, batch is
- PA - if you have your batch transcend states and lines, tagging errors to non errors
- JB - data quality checks record by record, if exceed %/proportion bad collection, may have stats on where errors came from, not sure to enforce pre-sorting of data
- PA - if i have set of data (NC and SC) and I submitted should machine render 2 sept batches
- KS - can we do it without calling new batch? return errors "AZ so these records dont pass" and have choice to fix, split and fix, etc. - doesn't invalidate batch to have one part of it messed up
- JB - idea - check record for format, some point too many errors, problem, if collex are from 2 diff states then need better analytics on data quality, enhancement of data quality checks - make it as easy as possible for carriers to submit data
- JM - agree place to stage things, rule to be run - day 1 fix process is on carrier, bolting on fix UI wont break prior architecture - add data qual state column to design, edit package will respond "bad, good, warning" - only answering data quality not changes
- KS - does req control mech to release data to allow
- JM - happy path - gets data in staging area, normal process kick off the rules, give answers and return answers to scheduler, if value is past safety threshold, if passes, done, if fails fires off email
- KS - control DB? Job flow?
- JM - if everythng works it just works, if edit is fast run the load to HDS otherwise intervention event - fire email to team/ group notified there is a problem
- JM - if edits return "pass" you let the job flow - have to go on assumption 99% of time batches run and work, worried about fixit approach - ideal is und why things are happening, these batches should just flow
- KS - pipeline approach
- JM - fan of persistent stage, 3 major boxes of ETL integrating with staging, stat model we can all live with on day 1, believes we will check 10-12 EPs, day 1 should be stat model
- JM - keep adapter as small as it if, explain it

Tues Sept 20, 2022

- KS - talked yesterday about 3 options for member enterprise, (see diagram), tried to simplify it down where we don't have "fixes" in the first cut, need to talk about diff pieces and figure out the phasing, basic contention was no ability to fix at all and should be on back end, but have heard there are folks who will NEED some way to fix data just before reporting, while we dont support first pass find a place to do it, 3 paths show where this can happen: fix after in HDS or fix while in staging area/pipeline (2) or fix before hand in frontloaded process - good idea of whats going on in this box EXCEPT the Adapter
- KS - this is what we plan to install in enterprise of member, one of the tenets is we provide data privacy - could be hosted and priv maintained thru hosted node having properl controls, security is well defined nd agreed to by carriers and carriers not moving data out of their world - but when hosted it is obvious data is outside their walls - if we could do hosted it would be simple, stand up whole stack inside node and partners (Senofi, CHainyard, CapGemini) could host nodes and maintain - feedback from ND, some carriers really want to host the data, striving for with this blue box is a tech stack amenable to as many people as possible - TRV, Hartford, State Farm, etc. - want them to be able to stand up this stack in their world - real exp with carrier who ran up ahainst road blocks in tech stack, policies, etc BIG disagreements - more complicated tech stack, 4 major sets of tech, the lighter the stack to install inside carrier world the more likely to be accepted with as little variation as possible - blue box not intended to be perfect, it has to take into account factors of company policies, etc. - try to minimize and maintain integrity of transactions
- KS - moving forward, call back to whats been done before, always open to best ways to document architectures
- KS - phase - get into each box, figure out rough block diagrams of whats going on, at the adapter
    - Blue - member enterprise, Green box AAIS setup, red hosted node - still believe a hosted aspect to this makes sense, complicated tech, young tech, challenging to set up and config, targeted set of people with skills contrilling we will be better off - some day mature (1-button install), Fabric and Kubernetes and AWS has certain complexity we want to encapsulate
    - No issues running node at AAIS at this point, will need to interact with analytics node (currently at AAIS, doesn't have to be), any carrier nodes need to interact with carrier enterprise stack where data lives
    - Adapter will run extraction, defined in a way it can execute in member's world
    - another tenet not challenged, the RAW data is in HDS, the input to the extraction engine, where member is agreeable the processed data can leave their world, into analytics, turned into reports
- JB - improvements on horizon for Fabric/Kubernetes (Fabric Operator), move in that direction, carriers are using AWS, lot of it using TCP/IP to make connections to resources, one set of permissions internally or access adapter or apps to talk to fabric, they will be interactions that req auth for use, ways to use it - thing that runs adapter that interprets request to get data would be binding approach: request comes in, not necc executable code but standards for implementation in HDS, concept of adapter good concept, needs to be somethign that happens when request received, hope there will be things that simplify the config
- KS - moving target, as much as possible, encapsulate that movement, other stuff doesn't break, a buffer of implementation and all that is trade-off, whats in the adapter
- JB lot depends on whats the db of the HDS and what will take, will determine HOW you get data from it
- KS - walk thru couple of different db types, see how they might make things harder or easier, weigh options, pick short/long term opps
- KS - we have HDS
    - noSQL (like Mongo, which 1 carrier said no to)

    - Relational DB
    - format of the data, allow scripting or programming
- JB - common denominator, what is supportable in carrier's domain, some form of standard sql, ought to support something common
- KS - current assumptions
    - Adapter
        - JB - interact with requests
            - carriers need to see and consent to requests, number of API calls
            - if hosted node wraps interface thru network, api call?
            - adhoc data call - do they know they have that data? (stat plans they know, repeatable), but adhos "what is the nature of the request", some knowledge of whats in the HDS? subsequent phase?
        - KS - function is to execute an extraction against HDS and return results (to analytics node? etc.)
            - if adapter makes as few assumptions as possible - needs to know format of db, cannot be just an api call
            - management among carriers will be handled via Fabric and hosted node, all the adapter does is execute the extraction upon request
            - carriers interacting with network thru hosted node, minimize dependencies, all the adapter needs to know "asked to extract data from here to there"
            - able to test the extraction pattern to see what it would result in
            - extraction should be human readable/understandable
            - some test harness "I have a new EP I want to run, test it"
            - needs to execute when data call happens but also eval before consent (can we do this?), flow says "I will consent but what does it do?"
    - db could be nosql or relational
        - sql? nosql? allow scripting?
    - extraction must be stored on-ledger
        - meta data, architecture of the request
            - sql? elastic search form?
    - extraction options
        - pure sql
        - scripting
        - hll
        - dsl
        - graphql
    - data model is (semantically) starting from notion it is the stat plans (transactionally based
        - PA - premium transaction and loss transaction
    - JB - currently stat plans, may evolve over time, starting point
    - PA - what are we adapting to?
- KS - Extraction Processor
    - receive requests
    - interpret / translate extraction
    - execute transaction
    - gather results
    - return results
- KS - big concern about security in this model
    - not sure if carriers have pushed back, passing around code which is gen the thing security folks say is a no-no
    - dont pass code that will run on someone else's machine
    - JB - push or pull model, automatic = exposure but if it is a request (pull) API pulls and evaluates it, not executing code it is interpreting, more secure approach polling for things to do and launching via human control
    - KS - where ledger provides some comfort, immutable ledger is good, seen code coming across, good with executing that, everyone has option to consent, some sort of technical way to test Extract process,
    - JM - agree with concern, intro construct to reject queries
        - if we had a table that said "if these keywords appear we will reeject query" - tricky to engineer but somethign that can screen filters, config table, system will reject if person approves, will submit request to improve security
    - KS - might not need but good to have, would run the initial request thru same scanner
    - JB - something validation of request would do
    - JM - humans make mistakes, double up on safety, hard to sell idea of arbitrary code execution
    - JB - late binding, execute as a way to test against what is in data store
    - KS - scanner validator, called as part of extraction processor, will happen via creator of EP, upload, system will run it, standard set of checks
    - JB - good add
- KS - elephant in the room - still DB decision, data format really tells us a lot of what EP is capable of, depending on reqs some may not be up to the task, take phased approach not starting with simple problem
- PA - processing with loops and stuff, stored procedures
- JB. - stages or phases with mongo, dont have to do it in all one SQL statement
- KS - Mongo stages of agg pipeline? Mongo proprietary, steer away from prop implementations (like map-reduce)
- JB - reason to not comm request in executable terms, implement in data store, one thing not in the current transaction data of stat reporting dont have the actual policy record itself, relational rep in the future would support
- KS - we do have poilicy identifier in datastore, could be business key to a logical policy record
- JB - dont have attributes of policy in terms or coverage
- KS - imperfect and difficult to put together, not all reports are policy anyway, agg of coverage across all policies
- JB - talked about extracting what fields avail in stat record
- KS - assume relational db and our EP was a pipeline of SQLs, one after next, no definition of statements (1, 2, 3) and EP executes pipeline of SQLs
- JB - all specification of how complex a request
- PA - likes
- KS - 1 not making up a language, not depending on someone looking at lang pre-process and und what it is doing, it is easy to execute, just clear text
- JM -
    - 1 can we create views as part of execution
    - fundamentally, is our problem a document store prob OR an entity relationship problem

    - Mongo is for doc stores, feels like a relational problem, insurance entites well defined and consistent
    - doc store not nature of what we are doing
    - dealing with well defined entities that relate to each other constantly, at 100k feet a relational problem
- KS - ready to commit to relational, fully supports what we need to do, aligns well with what we are after, transactional records
- JM - supportability - feedback from friday call, whole area comfortable with SQL, step away a more scarce skillset
- KS - document db supports simple model, outstrip it quick but simple way (2 tools - relational db and sql)
- JM - can we define arbitrary views
- JB - views used in staging, define and access, dont have to run first stage of pipeling
- KS - can we create views or use multi stage?
- JM - WITH clause in SQL, how complex do you want to get?
- KS - where the work is, battle b/w devs and db people, if i have to call you every time i want a new field take off, if we have to deploy a new db model every time we want a new view ?
- JM - schema evolution question, need to keep schema updated,
- KS - issue with old queries working
- JM - unless you validate JSON struct AND schema, no magic to schema management
- JB - might not change that often, 1-2 times a year at most

Monday 9/26/2022

- Member's Enterprise Options 1-3 (review of drawing)
- JB - push or pull?
- KS - pull is in the openIDL as a Service (orange box)
- Jb - polling for requests?
- KS - Adapter Tab in drawing
    - API external facing (listening for requests), gets requests w/ Extract logic (doc passed in), Extraction by Extract Processor, into DB we choose, EP is testable, before someone consents, eval valid requests, what returned, some kind of Scanner/Validator
    - Area - execution of code (arbitrary?), well known pattern not supposed to send code across and run it - make sure it will be ok, somehting that needs ot happen to that code beforre it can run
    - right now - map-reduce (moving away from), because proprietary to mongo and limited, if relational pipeline of queries, graphQL, get to point of assumption
- KS Member Node
    - complete processing of particular node
    - KS - Responsible for:
        - managing the network (Fabric) for a carrier
        - all of the interaction with the network
        - running chaincode
        - managing the ledger
    - DR - thing that takes extracted data and joins it with other data - function - acc to funct reqs
    - JB - other data, some might be avail on carrier side but not shared, could have universal data, but carrier info, granular and rolled up before sharing
    - DR - defined by EP
    - KS - extraction can request external data
    - DR - whatever comes back comes back
    - JB - ancillary data, in the node or carrier, sencitivity of connection - if you have data related to cust accounts, didnt want to share, addresses, etc., rolled up into info not shared (is it in a flood zone)
    - KS - part that makes it possible
    - DR - pink node, some data from carrier, take it, getting it to point where it can be agg and anon - most likely allready agg at this point, takes data, does intended JOINS, point where combined with other carriers data
    - JB - thought pink one of the nodes sending data priv to agg node
    - DR - exactly - the bridge, takes data from carrier and sends to other place
    - KS - analytics node is the "other place"
    - DR - go between
    - JB - place whree you manage req, see them in UI
    - DR - yes but housekeeping to the biz function - this ghets it there in a permissioned way
    - JB -where config (setting up channels, so forth) contained in hosted node AAS
    - DR - implementation specific, not business function
    - KS - configs the path to the network
    - DR = could be stateless too in theory - not a network
    - JB - not just data extract, if it. comes over api or logged into hosted node remais to be seen
    - KS - if we target the level of "gets data, moves it on", need to make decisions at some point whether in first pass mention Fabric or stay above,
    - JB - fabric for now, point is theres a control flow and data flow
    - JB - control interactions and data interactions
    - KS - node knows when to request results, decides when/initiates request for carrier Extract results
    - JB - after req consent - "Can you do this?" precede (what I calll control flow)
    - KS - somethign else, - this part of the arch is completely data agnostic, it is control, understands the workflow, does not care about what data is moved around facilitates the data, serialized set of data and pass it along
    - JB - data format agnistic because data is serialized, wotn care b/c it will be an opaque object
    - KS - would we want data encrypted as it moves thru?
    - JB - could be, doesnt have to be, need is another policy decision, if you feel connex is secure, wont be looked at by some router, then no need but it could
    - KS - could be per data call decision
    - JB - if encrypted need key management of data
    - KS - similar discussion with carrier from ND
    - JB - fine to do it, info as well as commms security, est end to end priv and pub keys
    - KS - what else? high level funct resp
    - JB - UI in that box would look at world state, see what requests out there, do it not necc automated, managed requests that come in (human), some type of login to hosted node to access UI - should UI exist w/in carrier? no keep it in hosted node to use the UI
    - KS - UI hosted here in AAS,
    - JB - need to log in, need access controls,

- KS - only allowing 1 org here?
- JB - make the most of the privacy aspects of this, hosted node assoc with carrier
- KS - as far as Fabric goes, one organization
- JB - still assoc w/ carrier, modular funct, security and privacy of the channel
- KS - privatre only to the carrier, permissions could be granted to other orgs if carrier decided, access to node is controlled by carrier - hosted node not plugged into cloud,
- JB - IP Addresses and ports - might be worth considering if carrier had people signing on and logging in, could be known to openIDL "who" so we can monitor hosted nodes (access not content) - some monitoring, logging in - simplify access capabilities, use as proxies for permissions to do work in the UI, log in and have cred in the UI as well
- KS - 2 layers of permission: manage node and UI
- JB - diff individulas who use UI, access to hosted node use identities for access to the UI inside the node - no need for mult ids and creds - to be investigated
- KS - current tech stack we could start from, all the tech running would be in this node right now, lay those out as a starting point
- JB - great, preaching to the choir for use in the testnet
- KS: Tech Stack (current)
    - Fabric
    - Kubernetes
    - Node JS / angular
    - AWS
        - mult services)
- KS - it being hosted solution, how much opinion will a carrier want to have about that stack? how would you think you would make your opinions known/how should we govern that stack? data goes thru, do you want to know all? need to know all? any idea HOW it should be governed?
- PA - Carrier X wants to know all
- JB - should be common across all nodes
- KS - what led us to this sep of concerns, does this tech stakc have to follow the policies of each carrier? knew it would run into problems? =- this is a tech stack decided on by the openIDL org, not any one carrier (contributed)
- JB - doesn't have to be part of the CTO standards as it is a hosted node, external interface, never get a stack that agrees with every orgs internal reqs - get agreement
- KS - goes thru the usual gates
- DR - security part they care deeply about, how it functs less so, security is big, had to come open on the stack, long way to build security trust, needs a lot of love for them to every be comfortable, startign from position of "not confident" instead of agnostic
- PA - Carrier X - very into being involved, want to und eveyrthing they are running and connected with, auditability, they are interested in knowing - believe a carrier was "how do you delete, how prove? where does data live at rest? transmitted, positive confimration all data deleted at the end of each cycle - some connected with all aspects
- DH - biz perspective, anything in the box abides by reqs established
- KS - 3 things possible - main concerns, tech stack is a big concern INSIDE, if hosted it is slightly less, very concerned abotu Security and the Privacy of the data
- DR - COST and long term sustainability, important
- KS - while might not have desire to control everything, secure, priv maintained, reaSONABLE cost, governance process
- JB - DH's point about lifecycle of the data, state of the data, keep in mind when data transmit from this node to analytics, uses priv channel, those things written to the cain itself, lifecycle management of the data
- DR - chain is only a piece of it, nothing stopping it from being written from PDC and elslewhere
- JB - PDC is only visible to partner at the other end
- DR - second leaves their node, could be written other places, not a perfect tech solution, good tech solutions, lot of this will be process based - legal agreements, auditability, enhance the tech
- KS - carrier in ND very concerned with this exactly, data sure where it will go, what you said you would do actually happened - visible, auditabl, enforced
- JB - benefits deploy in open source, can be audited as well as the logs, ways to achieve agreement with those convenants
- DR - OSS classic legal doc dump, pretty hard to systemically always check everything, still needs other peices and recourse and audit mechanisms
- JB - which code delployed, procedures can help see whats being used,
- DR - wouldn't over-index on that, 10 year old code has flaws no one knew, def not going to be primacy mechanism
- KS - combo of all those factors, agreements between entities
- PA - stuff we needed, tenets?
- KS
    1. must pass sec standards of all carriers
    2. must meet data privacy expectations of all carriers
    3. cost of running the code is sustainable
- DR - costs should be similar to API costs, not a $1, but commensurate with other services, reasonable
- PA - API is little different, this is data exchange plus cert of data w/ network, central auth validation
- DR - if you consider cost normally paid to stat agent, the tech itself shold not be near that cost - the totality, holistic cost, needs to be better than current state
- KS - stay out of value statement here, overall value may cost more but data priv, might be more important
- JB - bigger pict, efficiencies costs less, fewwer people doing bespoke reports
- DR - still in line with what we expect system to be, maybe problematic at scale, be cog we are building lean infra that is appealing to people
- JB - def keep that analysis in mind, overhead to monitor network, hosted nodes
- DR - watching to keep costs down, managed network is always more costly
- JB - small management staff, low overhead, keep it that way, point in the same direction
- KS - other types of nodes and their responsibilieis, analytics and multi-tenant
- DR - agreement the UI here is extracted data, where carrier resp ends, sounds like confident in tech stack, build this piece, mock up a dataset that fits what we think the EP would be, put it thru the pipes here, next step
- PA - working with Dale, can produce 3 diff sets of test data modifying what he has now,
- DR - stick it thru th plumbing, check the security, hands on keyboard and progress while we look at Extract Engine
- KS - analuytics node - still part of tthis side of the API call, stack for this network, if you have confidence in how that owuld work, attacking that, part of this - go offline and illustrate or do it tomorrow morning, same level, all comfortable, defining the projects to build out the carrier and adapted
- JB - along the lines of making stuff work, area needs thinking, what is the nature of that EP request, in the case of stat data "gimme"

- DR - start with what Peter has, see how it works, idea of what it looks like when done,
- PA - can do 3 json files as 3 mock carriers
- DR - if that confident with NaaS, can point to progress, the EP is the hard part and work on this in parallel
- JB - decision there, when transmitting data across interface, what are the handshakes? serialized, encrypted,
- DR - no persistent listeners, async responses to requests
- DR bigger issues are OPS and SLA maintenance, easier to be responsive than actively polling

27 Sept 2022

- KS: Recap of 9/26 discussion
- PA - stat plan data comes in, 97 characters, using stat plan handbook, converting coded message, (peter shows data model)
- KS - data models, intake model (stat plan - dont want to alter what carriers HAVE to send, flat text based stat plan when it comes in), stuff to the left
- DR - do it the right way, all pieces could change over time
- PA - not running EP against coded message, keep codes so if you need to do a join
- DR - trust PA to set it up, if he has an idea, assume right answer and move onto pieces we dont have a clear idea on
- PA - ingestion format is stat plan, HDS format is Decoded
    - for relational
        - 1 record per loss and premium
- JM - relational db?
- PA - totally put it in a Relational DB, suggest 1 table per stat plan, dont need to do weird stuff with mult values per column
- KS - claims and prem as well?
- PA - 2 tables per line, ea stat plan has prem format and claim format, main thought utilizing SQL lets biz users to contribute more (Mongo is tricky)
- **KS - make decision it is relational**
- PA - internal project, if switching to relational has significant implications, when do we want to put that to a formal vote, unpack implications, right now plan is debug first part of auto report, pivot to EPs working with postgresSQL, his team can pivot
- DR no strong opinon on which format but doesnt want to throw out work
- PA - most of the work has been und how the math behind the report and connect w/ business - has spent time learning JS and Mongo, finish up test with Mongo, can design tables and write EP after
- KS - internal planning discussion, JZ wants progress, prove stuff, healthier method
- DR - if saying "lets alter" - wants to make sure reasoning is strong, fine with Mongo
- JM - showstopper if we dont , legion of people using SQL, need to read EPs, team that manages and processes have deep SQL knowl and coverage but no JSON, biz unit comfort seeing relational form of data, ops team who will support forever very comfortable w/ SQL, can't do Mongo
- DR - use Mongo enough
- PA - into relational the whole time, wanted to see it thru with mongo
- DR - TRV can work with SQL
- JB - decide internally, move to relational
- DR - want peter to get resources
- PA - 2 fridays from now, wants to present THEN rewrite stat plans in Postgres
- DR - like postgres (can do JSON blobs directly) - what version?
- PA - still considering
- DR - Which version JSON blog functionality? find out
- JM - postgres favored flavor too
- DR - if we do SQL postgres is worth it
- KS - vetting internally (AAIS), impacts workstreams
- DR - ACTIOn - get Hanover opinion on this stage
- JM - denormalized? keep it flat?
- DR - like that too
- PA - def for the raw tables
- Jm -= challenge of flat, redundancy of data
- DR - performance in terms of speed or data minimization not a big issue here
- DR - node as a service, Peter free range
- JM - EP did nothing but SQL in postgres, highly normalized model, minimal connection, plumbing working
- PA - easiest over all, need stored procedures
- KS - why?
- PA - StoredProcs, do EPs in one lang
- KS - pass in on API call, cant do with stored Procs
- DR - **could** if you sent in API call and EP and stored as SP and...
- PA - Stored Proc - create and destroy temp views, looping, cursors,
- KS - stored procs are utility functs not part of EP itself?
- JM - zoom out - specifics - what auth do we assume we have over DB? Data management people strict
- DR asking for Vers control over DB, what permissions needed, manual requests? 3rd party agent?
- KS - if we assume only SQL, someone screams we address it
- JM - artifact now - table - can I create/replace (not exactly CRUD), limited # - tables, views, stored procs, etc. - then assume we have the right to create these? EPs can create on the fly? only a table of 7-8 objects, make the grid now - tables and we make em in advance - stored procs on the fly? bigger ask, but have it in this grid
    a. Objects as rows: Tables, Views, Stored Procedures, Functions, Triggers.
       What you can do with them: Create in advance, Alter on the fly.
- DR - look at as optimizations, utility helper functs? get it working with raw queries, I like stored procs - sanitize query, known responsees, good things about them, dont need them RIGHT NOW
- PA - better at SQL than Mongo, depends on how use analyutics node to combine the data, fair bit of process
- DR - do I think we will allow Stored Procs be written? No - dont assume initially
- PA - mid level processing layer, or whole thing in SQL
- DR - willneed midlevel processing long term
- JM - EP needs arb code, third party datasets
- DR - another thing it does help, simpler to maintain vers control of mid level middleware, some funct/procs than DB itself - this way DB is just datastore, need to keep it fed, moves vers control out a little bit, logical separation
- JM as little code in DB as possible, part of challenge - writing code in DB, put it in ETL layer - ETL at Query Time - if you find yourself writing code (stored procs) to get data out..
- KS - reality, people put in EP, thats what they can do in the next year, any updates to StoredProcs, will take forever to get approved by eveyrone

- DR - extensibility of Stored Procs is weaker
- Jm - once DB knows you are selecting, can't drop table in the middle whereas you can with Stored Procs - you can't screw up a db in a select clause but stored proc could blow up db
- PA - use middle layer, rest in JS, middle layer in JS and Postgres?
- DR - no pref
- KS - can we not do this with SQL, can't do this with layers of SQL
- JM - views are a logic layer, opp to put v type layer, dont want to overnormalize, can get rid of redundancy w/ aggregate functions, do you restructure your model, query layer or model in need of adjustment
- KS - governance model and bility to run code that has to be reviewed by people not just SQL knowl, optimize for somethign to be run by people ok'ing something coming across the wire, run into a lot of ugly SQL
- DR - sec architects, they will look at "where is my trust plane", arch whole environ to be secure and only expose DB, only run select statements against DB, more likely to approve something (if something funcky happens outside trust plane, can still protect data easily and will help), if "trust plane is behind DB and expose Data to some procedural code they will raise the bar
- PA - instead of everything in one lang and sql,
- DR - unfortunate process-gated thing, not the ideal engineering solution but where we are stuck
- PA - great discussion, how they look at sec, finish in mongo and do it in postgres, no stored procs, JS to make multiple select queries
- JM - dont mind scary looking sql, run mult subqueries and stitch together at bottom,
- PA - depends on who the person is reading it, simple objects WITH clause gets harder
- JM - do we put VIEW layer on here? is my model correct, team is adamant about keeping flat but can read long sql
- DR - plan there, may or may not need ot be middle layer, assume we dont need and only add if PA says he needs it? or assume we need it
- PA - middle layer to enhance EP?
- DR - third party data enrichment
- PA - entire report puts together,
- KS - assume we haver reference data nearby or not true? all transactional data?
- PA - state names replicate, state codes replicate
- JB - consolidation side
- JM - reporting logic, shouldnt occur here, not in the adapter -
- DR - only getting data over the wire to the servicew
- KS - no PII raw data over the wire, some agg happening here
- JB - cert have codes and dereference with labels later
- **JM - where does reference data get resolved?**
- PA - sourcing ref data from stat handboiok, on load, human readable when going in ETL: dale submits data, stat records go thru ETL, loaded into Relational data store, take code and add column for RI
- JB - in order to be human readable by carrier?
- KS - less Human readable and more out of AAIS codes and into
- JB - standard iso codes better than labels
- KS - bring it into something more standard
- PA - need to look, believe state codes are NAIC codes
- DR - we now have a plan for DB, plan for what happens after adapter done, unless we need it, role of adapter = accept EP,
- JM - long term design issue - if we do enrichment stuff at the grain of native data?
- KS - a lot of times it will be, have to be able to support it
- JM worried that if you run relational query, rule/val prop - run on fine grain data but arb logic from other data sets needs access to fine grained data, then therefore you extract priv data from RelDB, then filter it down
- DR - 2 trust planes
    - SQL first
    - Node - as long as it happens on the node, technically palatable, depends on how implementation looks, do we need quarantine zone, middle layer
- JM - opp to review data set NOT JUST EP
- DR - biggest design challenge:
- JM - human interface, there needs a gate somewhere that says "give a human ability to review data before it goes out the door
- DR - still one environ, if passing non agg data over the first trust plane, need secondary stop somewhere
- JM - "no humans" is not good, needs review
- KS - intention - testable dry run in final product
- JM - MVP 1 - dont put it but final product
- DR - same page with James, how to make it secure, one thing queries, will require executing foreign code in a runtime, not written by us, more powerful than SQL query
- JM - two trust planes, will be post-db
- JB - run some carrier side? access to the data
- DR - all on the carrier side, wont be in the node, but thing is the hard line in the sand for Sec - no code execution behind the trust plane (sql queries fine)
- JM - core deliverables - deck for the DB teams of large companies - selling to security people, will be work
- DR - most val artifact that comes out of this, JM/DR tell us what to say
- KS - cant accept solution without this box, they can run it or see results, arbitrary code only SQL (read only)
- JB - worry - some pattern that requires raw level, in the output
- JM - do high risk stuff all the time, but work at it (sharding, encrypting, jump thru hoops)
- DR - homomorphic encrypt for all? (laughs)
- KS - concern, are we back in "TRV does this, HRT does that?"
- JM - need to prove no way to do it simpler, hard fight but make the business case
- DR - always an exception process for sec finding, make as simple as possible b/c variances between teams, more and less strict, avoid needing to ask
- KS - not all will sign on to the risk or have the will to review it
- DR - inconsistency, one carrier agrees to somethign another thinks is risky, will run into, w/in a carrier depends on who reviews case and which CIO (who makes risk call), w/in carrier dont have one standard, decision makers still use judgement (there are still no gos)

10/3/2022

- JM - Wed Hartford has 2 new people joining effort, leadership allocated, JM still involved, new person on Arch WG, asked for build team for Q1, Q4 is ramp up time
- KS - talked to AAIS about redirecting PA from Mongo to SQL, AAIS is on board, whatever PA is working on should move towards stated direction
- KS - relational DB, run queries that are coming from community, talked about scanner

- KS - across API layer, all running in carrier, no side effect code like SQL, relational DB, allowed to execute against it to return response, deferring scanner and enrichment - did we say defer test facility
- JM - out the door with smallest MVP we can, solid like is core, dotted p2
- KS - ETL, submit stat plan, ETL turns into relational structure, EP Processeor executes SQL, across API and API returns results of that
- JM - is this good enough to turn into diagramn to show that, MVP: hand nothinjg but sql string over interface, Extract Processor, run it get it back, hand it back to interface have end or end plimbing - this is the core
- KS - nuance to execution of SQL, might be more than one sql? pipeline? worth discussin g now? or hope it can all be done in one sql
- JB - script? series of sql statements
- JM - agree it needs to support multiple queries, how do they communicate? pipelines - how do you get tem to talk to each other?
- JB - temporary tables, not modifying (create/destroy) is safer, in addition need to wrap initial investigation of request, step before
- JM - is validator on this side or the other side?
- KS - put it on both, create SQL, validate on the way in, if SQL dont know how it could
- JB - do need to investigate SQL - stat plan can say "gimme report", but other things will require you look at the SQL
- KS - going back to MVP, scanner/validator is out of scope,
- JB - if all doing now is agreening getting data from stat data, all SQL will addrsss, then scanner/validator do somethign simple, no need for consent dialog until we do the basic stuff
- JM - MVP in loosest sense of word or product out the door - can't do product until scanner/validator is done
- KS - MVP or POC?
- JM - sequennce the lines, 1-4 proves it works, 5, 6, 7 go to the industry - if solid line is #1, what should second be - more about proving own assumptions/proving value to industry?
- DH - business perspective, security from going from TRV node thru adapter to analytics node, showing data privacy
- KS - lot og reqs, go and pick the ones that drive second level of POC, things required and implemented before we go to production
- JM - enrichment 4th on my list, of business-y things, scanner-validator says "ok?", test validation says "run 100s of rules to prove it is what you think" or Enrichment?
- DH - prove my data is protected
- KS - the scanner - #s 1 and 2 show security, DH wants to show rest of the flow and across nodes we make sure the stuff is going the right way over analyutics node
- DH - basic plumbing
- JM - end to end plumbing? least I need to do to prove it
- KS - briunging an arch perspective here, 1st thing: does the plumbing work, 2nd can we install in a carrier - imp it will be acceptable in a consistent way across carriers - what we work on here can abstract or work concretely so everyone can run it
- JM - Enrichment is scariest
- KS - needs robust plugin capability or external data model - trust and maintenance are hard, diff timeline than EPs
- JM - stop at dotted line and focus on solid lines, all we are gonna do
- KS - step 1 solid lines, step 2 end to end plumbing, step 3 up for discussion but for KS "does this work in yoru carrier node"
- JM - focus on solid line
- JM - get mult sql problem, need mult sqls, couple ways to solve this, argue this executes out of schema with no data in it, schema that carries data, give s a level of grant writes to DBA team - ask what is the set of writes asking for in sub schema
- KS - can we est a sandbox DB where the EP works, updats, creates tables as necc
- JM - can say that is a design principle and allow implementor to do either - golden -
- KS - sep schema
- JM - pros and cons of each, takes fight out of DBAs
- HDS Schema, EP Schema
- JM - separating easier to ask for things - default: want to create views on the fly, idea of mult sqls, parsing behaviors, smart enough, put it anyway
- KS - could be a collection of strings
- JM - how interacting with each other
- KS - first can retunr, second consume, you test it - sql string updates intermediate table with data, all yo uhave is a coreogrqapher (run first, point second at results)
- JM - intermediate table, SQLs comm with each other, creation of complimentary tables - DBAs protect data - ask for "Create Views" auth OR temporary tables (sometime dont have robust you want), ony talking about views, temp tables or phys tables
- JB - creation of indicies
- JM - assumed, agree not to use Stored Procs, comfortable saying "want schema, in schema have these rights, agree to flush data (flushing policy)
- KS - drop the whole DB
- JB - temp tables when connection closed by default, but drop statements good to do
- JM - might be worth asking for matrialized views
- JB - temp tables for intermediate results
- KS - keep saying not time bound, if we have to do more X not a prob for performance
- JM - if this is a long list we have to ask DBAs for
- KS - assume "yeah", go ahead and create schema, inside POC, extract processor remove/create tables
- JM - grant grant grant - pretty minimal,
- KS - table creates and stuff
- JM - phys table question on there, more pushback, flushed data is question mark
- JM might find optimization opportunities
- JB - consistency issues, better to treat like workspace and flush it
- JM - no phys tables for now, materialize views?
- KS -seems like EP has more complicated interaction diagram, interaction diag between 4 components?
- <Live diagramming>
- JM - view is a mech to make one sql depndent on another sql, materialize view "love the view but need performance", phys tables fix perf issue but need in advance - want to go as far as we can to
- KS - running first sql, get results, where kept?
- JM - temp table, self-destruct at end of session
- KS -DDL?
- JM - easier ask
- KS - go0ignm to have to describe the structure of results so temp table can hold em, has to be done before run first query
- JM - draw out one query problem - EP should have arrow to postgres SQL and say "SQL", assume it does, will then (logical not phys) will largely read from HDS, retrieve data, postgres to EP schema
- KS - somethign in SQL says "select x from this table " returns result, and persistning temp table?
- JM - retrive from HDS should be enough, in any non trivial case results go in temp table, if we assume we wrote temp table, the Extract Processor runs retrieve

04 Oct 2022

- KS - recap of 10/3 call
- KS - Adapter tab in Lucidchart
  - solid lines in first POC / MVP, proving point
  - dash lines phase 2 - prove end to end security and privacy
  - other MVPs are technology, suitable as a common implementation across carriers, lots of other requirements, executing some reports
- KS - get first couple phases defined, solid lines - decision wanted to make and bring to TSC on thurs, DB will be relational, most likely postgres, dont think phys db is what we will land on as a decision vs what level of sql capabilities it has, needs to be able to do these things, clarifying (exeucte DDL, lot of nuance, temp tables, etc.) - needs to run SQL and SQL queries in modern way, store data in json fields, some extensibility w /o schema migration - will bring to TSC on Thurs as
- PA - JSON in the row could create problems
- KS - not sure but worth discussion, revisit concerns when draw up decision, prob not part of first phase anyways, keep it as a thing to challenge
- KS - what is in first phase (boxes in and out, part of arch running on carrier/carrier footprint)
  - API interface
    - reach out via scheduling or polling or challenge/response (request from another service: "time to run an extraction"
    - resp for taking a request w/ EP on it
    - in EP:
      - SQL, other things we uncovered (next page)
      - parameters, allow to work with diff states
  - Extraction Processor
    - gets data from HDS
    - runs 1 or more SQL
      - to provide robust solution might need "pipeline-ish" sorta thing
      - shouldn't do any DDL against HDS, DBAs would be ok with this as loing ad DB is unassailed by side effects (new tables, create views, change of data by EP)
      - collection of SQLs executing, requires some intermediate spot where first sql goes, picked up by second (see sequence diagram)
      - first SQL gets results, to second SQL (requires second schema, allows for temp table to support results of first query)
      - Extract Processor -
        - result of SQL1 is another schema
        - "schema 1 results in this dataset, create temp table to hold it, run first query, store results in temp table, continue process to last SQL and results of last SQL returned
  - Submission Stat Plan
  - ETL process
  - HDS schema: flat, denormalized, relational doesn't have to be exactly stat plan, simpler read, not flat file
- KS: James has resources starting when? (Sean to find out), AAIS approved moving from Mongo to relational DBs
- PA - right now doing ETLs with significant amt of JSON, do we want to sep ETL from HDS?
- KS - sep conversation, not required to go to that depth, if Hartford want to help with implementation then gory details
- KS - should have with MVP working software to load data into HDS, process, get result - helps arch in the room w/ what is running on carrier will work, hint the stack is not so onerous it can't be implmented across carriers effectrively
- KS - 2nd MVP
  - could embed execution of stat report? original scope, get a stat report working w/ that plan format, way we show the full end to end plumbing is working, show that data priv maintained, see whwre things coming and going,
- KS - sequence diagram / interaction diagram (Adapter Interaction on Ludicchart)
  - Client (hosted node)
    - requests
  - API
    - responds, asks Extract Processor to execute EP, mult SQLsparameters, results schema
  - Extraction Processor (SQL)
    - as we execute for each SQL that comes in,
  - Postgres SQL
  - EP Schema
    - allows create temp tables or views, execute same retrieving
    - continue process creating results from each SQL, tells next SQL to use the new table
    - creates pipeline
    - last SQL retrieves from last temp table and end result goes back to client
- KS - seems safe, well known technologies, sounds like tech stack has no need for Fabric running here, could be network comms agnostic, reply to a request and execute code (JS?)
- PA - version of Linux in relation to Postgres? thinking what PA will do when he gets back, spin up AMI, bash script to install postgres, way he installs may be different
- KS - EC2 or Lambda - will see
- PA - maybe inside Kubernetes
- KS - Postgres could be hosted too
- PA - using more amazon specific stuff, linux box makes it cloud agnostic
- KS - would like DavidR or JamesM to weigh in on those decisions (suggestions or "we will configure as X"
- PA - general thought, big carrier who roll own solution should but small to midsized want easy to use test environ on their own
- JB - this is partly why we are saying lowest-common denominator SQL
- KS - least controversial technical stack is what we are after
- KS - this is enough for peter to start directing his work when back Monday, can't go deeper on decisions yet, big decision not useing NoSQL and committmeing to relational DB for HDS, bring to TSC thurs and get a vote, tacit approval from DavidR, JamesM
- PA - how about debugging of Mongo to show DMWG it works? continue? plan to present next friday
- JM - persist temp tables for the next SQL, could put "read temp table from EP schema" to be more complete, end of the day SQL code but the BEHAVIOR
- KS - first SQL, might jump straight thru and return results, if other SQL will persist w/ temp table and continue until don't have any more queries, return results, important: allowed to create temp tables on second schema where we can't do that on HDS (HDS is sacred, no side effects, cant change data, cant change schema) - do we want to write in the dropping of schema?
- JM LINK
- KS - every time we run SQL we return results

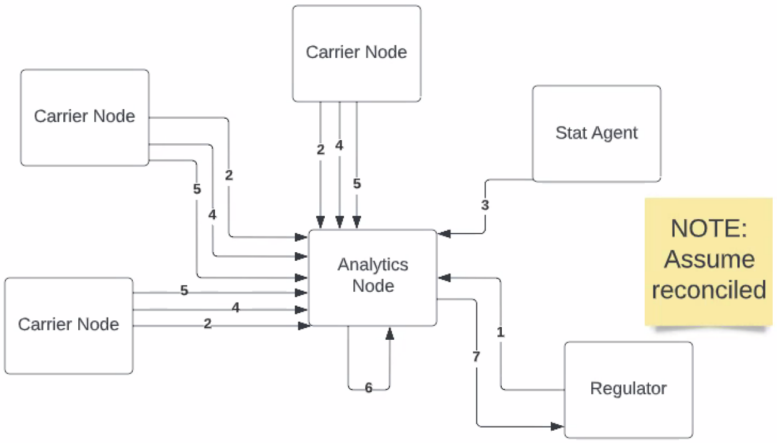
- JM - evident from diagram to the user the writer of an extract script, only the last SQL will result in results go back to client from API, useful to show in loopbox, that there is a looping behavior for any nymber of SQL for any number of tables
- KS - dropping the table at the end if it collapses you can debug it
- JM - serious implications
    - say 8 sql
    - first 7 make intermediate tables
    - 8 is the only one to make final results
    - purpose of intermediate tables is for processing
    - all data stays in EP schema but if we show results in the loop at processor level means the data is coming back to Extract processor
    - final select statement only one allowed to write back
    - intermediate results only live in EP schema
    - whole point, only get one result at end of day, designer of EP needs to know
    - must drop all after final retrieval query
    - simple case - one sql and be done with it? go for it
    - http://www.cs.sjsu.edu/~pearce/modules/lectures/uml/interaction/SequenceDiagrams.htm
    - first sql, last sql results
    - any SQL from #2 and beyond can read from prior temp tables
    - some reasonable degree of paralle processing
    - rules for writers of EPs
- JB - if results are result sets from last sql statement, flat structures, is there a case where we want to return something w/ structure, serialized result
- JM - if we modularize this write, in future release, you could put loop box around EP +, any one hit of the DB returns 1 table could you loop above (now EP can do mult scripts on mult datasets)
- KS - if you said "read all by zipcode" then "by floodzone" trhen "read together" - link by foreign key - not single flat wide row if each pieces is building more structured model
- JM - if sufficiently modularized once, should be able to do mult times - if you make a rule only last can do output is a functional req, has to have mechanism to know when popping out with results
- KS - last is always flat result set
- JM - throw on day-2 list, agreed denorm is bettrer than norm, etc. - denormalization argument
- KS - will pipeline of SQLs support these reports, sooner we can throw a real pipeline aghainst this model the better
- JM - in question, flat and wide aggregated, sorted, filtered - if mult sqls, EP can run any one against HDS, still wondering how it looks at attributed fine grained data,
- KS - allowing mult SQL to run against HDS (constraint: retrieve read-only queries, one or more tables, any other SQL could read any table created temporarily, eod have one query returned flat 2-dimensional result)
- JM - w/ that design, EP is firing off SQL but not seeing data, only sees final data - downside: mandates single flat table (could see situation where you want more than one table) - the Extract Process might want more than one table - if we say EP must be aware of more than one result set, needs function that allows EP to run any number of datasets, if we only allow EP to be the place to see where it all comes togethe then
    - 1 EP sees fine grained data
    - 2 will need Py/JS or something up, not SQL to read it
- Ks - unwind it
    - the final results could be multiple result sets but combo could also be done in that loop here inside the SQLs if you need to return mult results, EP never has to see fine grained stuff over here
- JM - hybrid
    - prescriptive
    - say have API and attach DOCS, some # of docs
    - allowed to call API w/ any nyumber of docs, in 5 docs is SQL statements, any nyumber of SQLs,
    - accountability so that only last one can serve result set

10 Oct 2022

- Introductions
- KS - TSC decisions, pushback - document arch well, keep track of decisions, only thing TSC needs to hear are things that are controversial or heavy weight (JM: big decision or controversial) - need to document the decisions well, consistently, get sent to archs in the room - lots of frameworks out there, happy to use one that works - what we did for what we have right now
- SEAN BOHAN TO GET MILIND AND MOHAN ACCESS TO LUCIDCHART
- PA - eval of postgres to gen the autocoverage report, state-by-state breakdown of auto ins in the state, spend time TUes meeting on how we do that eval, get aligned and going in a way we all thinks make semse
- JB - feedback from maybe Hanover
- Faheem - SQL server shop
- PA - doing open source stuff here, dont want to put AAIS in position where they need to pay for mult solutions
- JB - if carriers can use compatible sql platform...
- PA - for eval would like to use postgres and how that connects, every implementation of sql is different (setup, strings, lot of similarity), 6 months ago discussing need of mult options, test ex of all, do postgres first, how much we want to use
- PA - walkthrough of the platform
    - RDS vs EC2?
    - Milind - yes. why? maintenance
    - Faheem - better to use managed service than manage this yourself - is the blue box what the carrier maintaines or per carrier in AWS
- PA - blue box (diagram) - each will have their own, segregated account, isolated from rest of enterprise, your infra you will manage, hopefully get to high level of automation where it manages itself
- FZ - if we can as a group propose an automation that works well in AWS, more standardization, more debug, makes sense -
    - mostly azure shop, starting cloud journey, how much effects or not
- PA - Hanover is mostly azure, started few years ago - you could because of the adapter connecting into blue box, think to do something in serial-izable way, one could run azure, one could rune somethign else - our enterprise on AWS, more comfortable, initial trial would be idea in AWS
- FZ - are most on AWS?
- JB - should be able to run the network across clouds
- easiest for people to work with, run on azure, fave platform
- FZ - will help to see if it makes more sense for Hanover,
- JB - some of the work oriented towards platform as a service, more than one cloud
- PA - can explian w/ AWS but would like to learn equiv services (like azure) - been working with internal team to deploy existing arch, all in terraform, using service like terraform for deployment for multicloud stuff - using in this box - S3 buckets, serverless functions, bucket storage, S3

in the ETL, system of record outside of the blue box - stat record= string, dep on position or number means diff thing, one for each row, record generated from it - ETL= gen stat records, load into S3, trigger serverless and decode the record, from coded string into json - thnking RDS makes sense

- PA - not where we are exactly, is terraform what we want to use or somethign better?
- PA - premium transaction, policy booked, info about whats being insured - also have loss record, same length - doing it in mongo used one collection (collection_insurance) and used JSON, going forward with Postgres, thinking 1 table for prem transactions and one for loss transactions - nature of way data comes to us: flat tables that cat all this info in one rown
- MZ - not storing json in postscripts
- PA - one column for each attributes, maybe more columns, get stuff like like coverage code that might be "2" (bodily injury), put it in the db, putting coverage code and the human readable one - so we have all keys for joins and agrregations, read-in, loaded-in - going to need service-user ability to write load, sep service user for doing reporting vs ETL
- FZ - operationally keep it sep/secure - what portion of the flow outside of the blue box?
- PA - loading HDS and loading info about HDS, will be where your target for your stat data, regulator comes up with data call, blue box is carrier node
- PA - regulator makes data call, Analytics node works as orchestrator, carriers notified of data call and can like or ignore, after data call has been liked a Stat Agent adds Extract Pattern

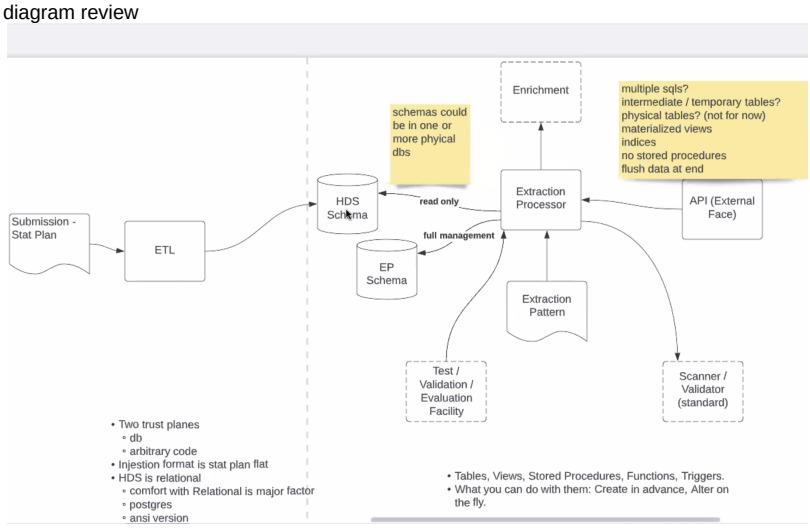


Process	
1	Regulator Creates Data Call
2	Carrier Like/Ignore Data call
3	Stat Agent adds Extraction Pattern
4	Carriers Consent/Ignore to send Data
5	Data is sent to Analytics Node
6	Analytics node combines data generates report
7	Report sent to regulator

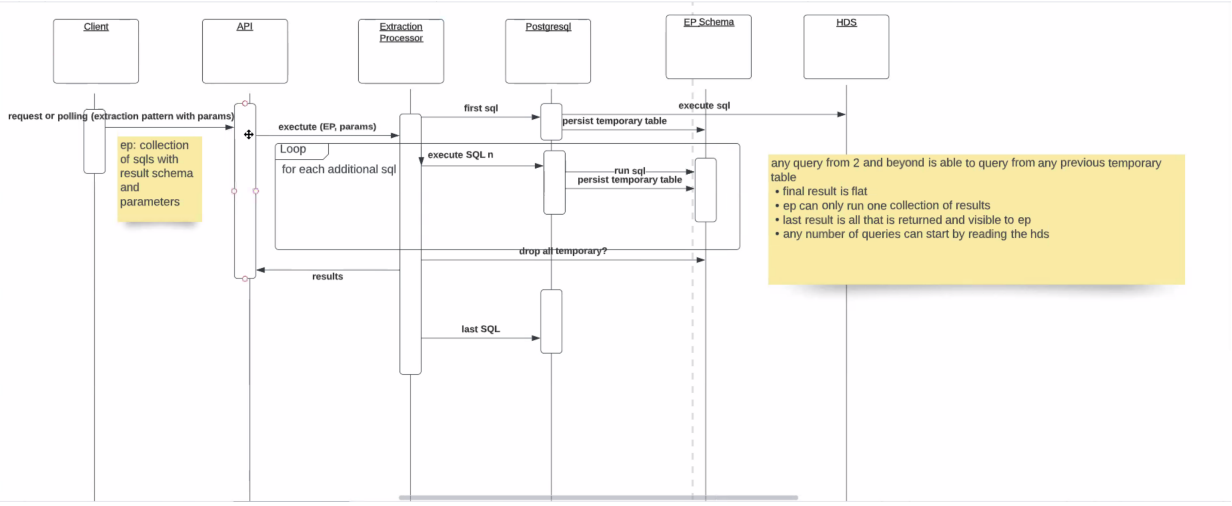
-
- perfect world all carriers send back data, data always aggregated

11 Oct 2022

- diagram review



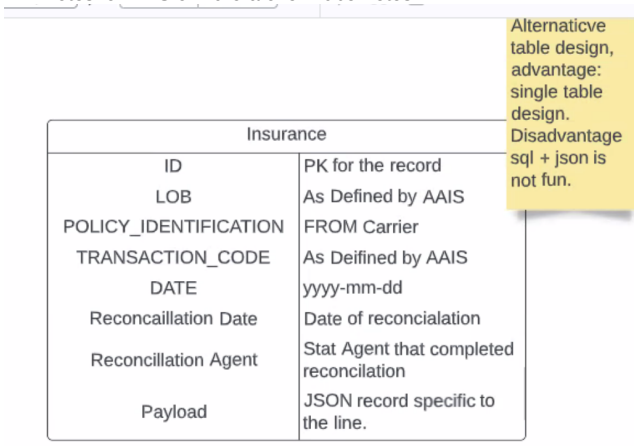
-



- DR - standardize on a SQL engine and store or more cleanly decouple the API from DB, more complicated but bespoke for each carrier - here is a paylod with what we need, standardizes whats avail in HDS, put it all on the carriers to do extract for api call payload OR standardize on an engine - in between solution is allow carrier to decide, API layer doing what it is doing now: passing sql payload, and if you choose to use DB prescribed by openIDL - but if a carrier wants to use another they can write their own inbetween - allow to move fast for this one
- KS - can we guarantee extraction is the same across carriers?
- DR - no less than today, if the HDS is the same across all carriers, intro risk EP written by carrier isn't accurate compared to EPs on nodes
- PA - current workflow the Stat Agent is writing EP, carrier consenting
- DR - writing same thing for mult engines harder - we standardize on engine, stand on API, most carrier just do that - in the corner case wherre comeone refuses or cannot they have option to do it another way and cert data is accurate
- DH - how work? with standard set of queries for stat reporting: fixed, known, unchanging yoy - in this case the EP is unique
- DR - standardize on the output, assume most carriers stand on whats made for them
- JB - ad hoc request relevant - efficiences, need ability to execute SQL, just b/c postfges doesn't mean postgres - create tables and views something a DBA may not want
- DR - an addition, "can we use mult engines?" not sure we can - need to use API extract layer, doesn't think we need to , more palatable to carrier, if rhey cant do data calls efficiently they can revert back to bespoke (their decision)
- DH - value goes away if we can do data calls efficiently
- DR - doesn't change how we interact, from exp there are carreirs who have to use X, if we support too many a problem too
- JB - analysis of what level of SQL is common/needed
- DR - not sure you will find most SQL engines that can use generic sql efectively
- KS - opp to discover
- DR - he is ok with this approach, but in resp to ? "lets expand db engines" - he doesn't think expanding is the way to go
- PA - use time to pivot to notes
- KS - 2 things
    - can we defer this decision - doing it in Postgres, ID where there are postgres specific things, ways there is a smaller footprint, und what it looks like, create table structures, do it in postgres, best choice, maybe add in mariaDB, other things
    - if we go down postgres, is there somethign we can articulate, or put into docs, "if you do this then..." have we thrown the baby out with the bath water asking people to write their own SQL
- DR - try to make too cloud agnostic, too many abstractions and compromises, go w/ postgres for now, not a bad thing if carriers have to write their own EPs rather than they dont join the platform
- PA - give DBA a postgres compliant EP, they could convert to another engine
- DR - and provide sample dataset to run both against
- KS - if allow them to write their own, give them a testset
- DR - preferred approach, go ahead with postgres, know how to handle this, dont entertain us maintaining us writing mult EPs for diff DBs
- PA - by halloween postgres working, test it against oracle stuff, will do analysis, syntax things, ID whats not compliant with others
- DR - James advocating this approach, looking at complexities - we are good, dont need to make decision now, dont believe if we have to expand access natively supporting EPs is the way to go, provide interface spec and allow carrier to cust as they see fit
- KS - keep efficiency and guarantee all are running the same thing
- DR - doesn't care about EP code, cares about the data model
- KS - diff perspective
- PA - PA HDS

- RDS
- 2 Tables Per Line
  - Premium
  - Loss
- 4 User Types
  - etl_user - loading/writing
  - reporting_user -reporting/read
  - developer - selecting
  - dba - DDL
- Table Design
  - Flat
  - Codes + Human Readable
- Index
  - LOB
  - TransactionCode
  - Date
- Location
  - openidl-main/openidl-hds

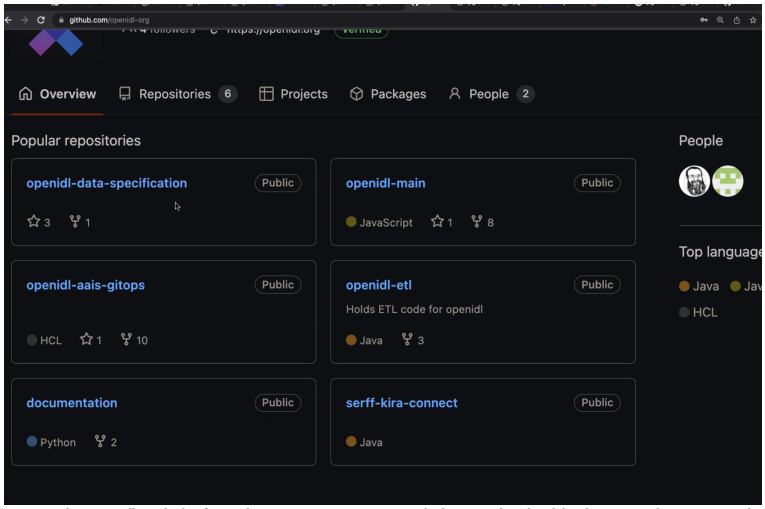
- DR - votes for RDS or Aurora over Kubernetes



- PA - payload specific to record type as JSON, simpler for loading but more complex for Extract (Sql/JSON not fun) - 2 tables per line (1 prem 1 loss), index tables on LOB/Tracnsact code/date, all scripting goes to main/openidl-hds
- KS - why not just have 3 tables? bunch of overlap between the two, not saying should but working in there, prob 10 or so fields all the same
- PA - accounting date on both of them
- KS - largely denormalized, transactional, single coverage per transact, can be flat, but some overlap between the two
- PA - 2 diff objects and one object per table
- KS - take the common stuff and put it into a 3rd table
- DR - doesn't need to be very performant
- PA - seeing two tables as raw tables, idea of easy to access business level at a later date
- DR - makes sense
- DH - loss divided between outstanding (incurred) and paid - diff transactions - transacrt of paid and incurred total amt expect to pay on that claim
- PA - use views to see those? working on trying to find the most recent outstandings, view layer may be something to look into
- DR - if we use views on day one, may be complex and relied upon
- PA - working under KISS
- DR - views are great, make things easier, sometimes become relied upon
- DH incurred inculed paid while outstanding is "whats left" in a point in time
- KS - can't und suggestion - give diff data?
- DR - lots of use, horrible data warehouses impossible to understand
- PA - use an anon block and do it in a single sql

- KS - script or statement? script, postgres specific
- MZ - I joined this week, so sorry if I am asking repeat questions. In DevOps wiki documentation I saw GIt and versioning strategy. Are we also planning to use similar for data base change and versioning, like using Liquibase? Schema changes?
- KS - too early, single schema, not for POC
- DR - asking for lots of maint and specilaization w/in db, might be tougher, goes to last point, once get more complex then more complex versioning and deployment (accross carriers)
- PA - make sql statements independent of sql scripts, get to maturity we can automate to track DDLs
- KS - looking at stat plans dont get much schema migration, big ask for someone to ask to change schema
- PA - depends how you do it? if we add more columns to end of table wont break existing
- DR - not change is simple or easy - it is a change
- JB - manage over network
- DR - less complex you make it for use to adhere to the more adoption you will get - value to TRV is extracting complexity from them, closer you get the better adoption, more it is a chore to maintain the less adoption and more resistance
- PA - last bullet point, ok w/ location for scripting? make a new dir in openIDL-main

17 Oct 2022

- bump into challenges and we will describe it
- KS - what should we be talkign about now ? structure of script? Peter? Talk about details of what David brought up (temp structures in postgres? multistep processing?)
- PA:
- 

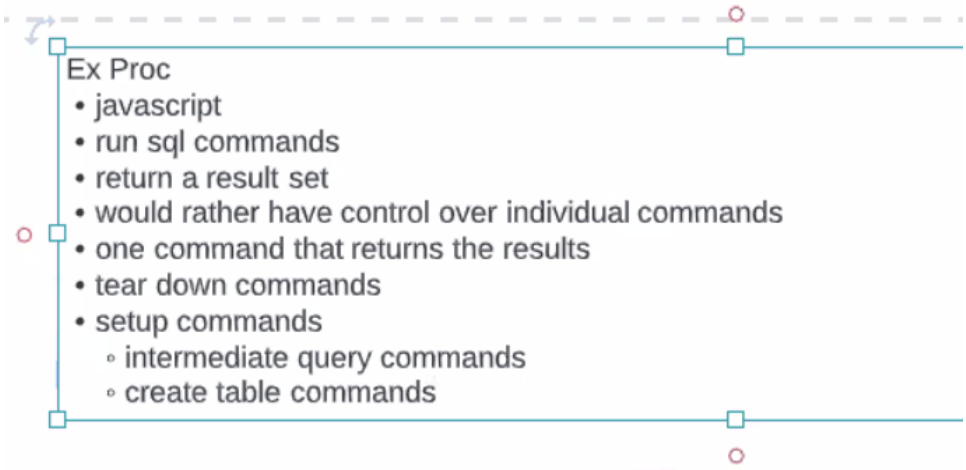
- PA - sub apps/lambda functions, new: openIDL-hds, worked with data engineer, produced table for prem records, 1 col per attribute, mostly varchars, mostly numerics - worked today on loading script, simple, not super fast, havent gotten into making the report yet, delay until we talk to James, brought up Friday, need to add col, unique ID to each record we receive - sequence or GUID? table until James is on the call, will faciliate testing - by next week will have loss and prem loaded, - https://github.com/openidl-org/openidl-main/tree/hds/openidl-hds
- PA - DH and PA looking at metrics and KPIs from data sets, good process
- KS - source data?
- PA - got a company closed, got records from 15 years ago, modified, encrypted and removed identifiers, 10k prem records sanitized and 5k loss records
- KS - source?
- PA - using coded messages - originnal stat plan formmated data, not edited, has been decoded
- DH -cREATED expected values based on the data provide, to test that we ult get it right
- KS - slightly diff than what we gave to DH
- KS - this is the HDS format?
- PA. - table create statement for premium record/auto: https://github.com/openidl-org/openidl-main/blob/hds/openidl-hds/AU_PREMIUM.sql
- KS - EP will run script against HDS to produce one carriers slice of the reslut that is the stat report, data from one carrier's perspective
- PA - earned and summed premiums,
- KS - col by col, needs to be a single script - haven't decided on managed postgres, etc. just that it IS Postgres
- KS - just postgres data source
- PA - once we start looking at how we deploy we look at how/what and then "how is the adapter going to connect", big points of decision
- PA - JS + SQL
- KS - not saying allowing JS to run as part of the extract, as a starting point
- PA - like Python, but chaincode...
- KS -chaincode is in GO, we dont touch, all the code for ui, etc. is in JS - not talking about Kubernetes at this point
- PA - for HDS, dont know enough about adapter or chaincode
- DR - dont think we need kub on carrier side, JS as a lang, py is preferable but dont need to, this is time to discuss "how do we anticipate that working": sql script, return data - what does that look like
- KS - if i were running inside AAIS direct, set up lambda with API gateway and that sit, not sure if it is right direction for this, huge decision, ok saying 'run as a lambda?"
- DR - think so, could also step further and say , the hard part is the SQL script, could extract to point where you say need API, need - api - people will take, lift/load and use it - how do we encapsulate that sql statement?
- KS - plaintext?
- JB - some form of protocol to say what request is to wrap that text, something to create table, sep from extraction, simple message/protocol
- DR - how much is predefined vs extensible - easiest way is dumb API, sends super simple encrypted sql statement, assumes prior knowl of what to expect? fine or need more extensible
- JB - some kind of header or content descriptor, not blindly execute
- DR - could make it more flexible or more hardcoded for lack of a better term - schema vs schemaless, well defined vs open ended
- KS - api results schema is json object or all these fields, assuming going in, one endpoint, having diff endpoint for ea use case wont scale well

- DR - quicker, but wont scale - other ? - is there a need to have a signature of this to verify against? executable code asked to be run - some check needed to check against "yes this is an API call with a query to execute how do I make sure this is the right query"
- DR = some mech by which you say "yes I have an API call and it is what we say it is", maybe built that into auth flow? payload is trusted? thinking what cybersec peopel would want - exec code that returns data - million ways we could do it - do we want somethign to verify the payload is unaltered - simplest a CRC checksum signature kind of thing - if you download my code this is the sig you should get - code wasnt tampered with or you did something goofy - notes it is "RIGHT" and makes sure all are running exact same code against db
- KS - trusting what they are asking is what should be run, high level
- DH - signature both ways? here is what we want you to run and here is what we ran?
- DR - outbound, assume it is all encrypted in transit, have some negotiated key exchange, know it came from us - some kind of handshake - nuance is in one direction there is a party executing code from an outside source, API off-flow, e
- FZ - if altering data, ID of sender and instructions are compatible with what you are sending, if ID and instruction not altering anyhting
- KS - James put forth it should not alter HDS, where we came up with second schema - need to know what that script is gonna do to run against the HDS a
- DR - quick easy check: here is what it is suppsed to do, here is the signature for the code that does X, verify
- JB - session based?
- DR - extra layer, when you run this checksum, code, hash, this is what you should see and if you dont see it something is wrong
- DH - how do we know code is not going to cause any harm to HDS or deleting records, etc.
- PA - compartmentalize the risk, doing reporting, specific users
- DR - down the road more important, running not only against the db, limiting connections
- PA - RDS has features, amount of risk isnt huge
- DR - not huge but also lightweight - hash and publish the hash before running, gives a sense of security and consistency everyone is running exact same code
- PA - signed payload like linux ISO, can pub hash, will have more insight next week, earned prem, small way to show what is req to get here, not too complex, help carriers going thru cyber audits
- KS - if one API vs mult schema oriented apis then generic schema
- DR - the other discussion, not all hash needs to be transmitted over the wire, pointers to docs that desc what it is, dont anticipate feature rich facility at the begininin
- KS - getting somehwere with actual script is the biggest in the near term
- DR - get something to show as soon as possible: API Call Made, Code Run, Results Returned - dont get bogged down day 1, things need to be there, can be handled by references - from carrier perspex that would be good, maybe info is pre-shared or published in advance to keep API logic down
- KS - if we can get James involved, would be good, talk about running the script and that process tomorrow (10/11), did start moving documentation on the wiki to make room for incoming documentation

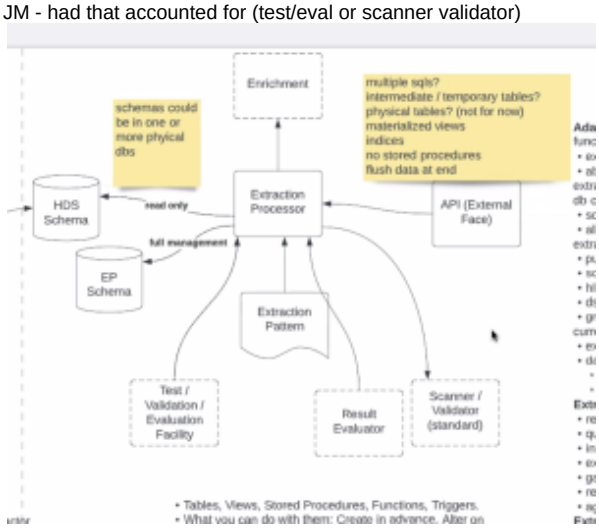
18 Oct 2022

- PA - (Takes us through the work he has been doing with the test dataset https://github.com/openidl-org/openidl-main/blob/hds/openidl-hds/AU_PREMIUM.sql
- PA - Sequence or GUID
- KS - unique identifiers - who owns the ID? does HDS own it or do we expect outside of HDS to do stuff w/ it
- PA - primary key on record at transact level should never leave your area/priv account/pdc, makes sense to have that decentralized and controlled by HDS
- KS - no other outside systems dependent on it either - for debugging
- JM - benefits of GUID - scalability, ease vs scale, interesting question of how postgres chokes at scale? do we care? put it in a GUID have more parallelism than choking in sequence, load on this is slow and dont mind sequencing or parallelization you are done - if you have a GUID generator throws a 36 byte string guaranteed to be unique and run massively parallel processing - only a scalability question but does complicate things
- PA - misunderstood guid - not a guid for every single carrier but across individual HDS
- KS - do not want to throw out massive parallelism
- JM - GUIDs unique as well, will we ever put together for mult carriers? No, not at transaction level - can we use SEQUENCES on day 1 and keep GUID in mind if you need it
- PA - Milind put the command in to do the guid, can make that happen
- KS - thing you lose with guid is ease of understanding of the key, takes time to work with the guid, has pros and cons, a dev staring this down another thing they need to keep in mind, if using primarily to debug stuff, tend to use GUIds for stuff and uniqueness
- JM - not massively over engineered, make sure seed it correctly,
- MZ - GUID is better 🙂
- KS - nothing says must do sequence OR guid, if not hard to create, sequence is unique, guids might be. worth it
- PA - insert 1 vs insert many, doing parallel stuff will be key
- JM - test the hell out of it - no magic, big fancy hashing algo, depends on what you hash: server name, etc.
- PA - will do some work on it
- KS - adapter and API layer, sequence diagram and lost on a couple spots, why do we need a second schema, cant DB do things w/o destruction to the schema
- KS: Guiding forces
    - transax processor runs a single sql script - create table-run query, etc.that can do mult steps w/o being destructive to HDS, having a second schema
    - how do you get results back to processor
    - sep schema is to appease policy reqs that not too much control over hds schema
        - split schema for org considerations
        - dont want higher power in HDS schema
- PA - stored proc (dead), maybe use anonymous block to have one sql script executed
- JM - talked about simple and ended in complex, the challenge is what gets back to EP, if you execute the sql script (fairly trivial to write big script) - issue is how to get input back to EP - make query as complicated as you want but only get one result back - ? - what if you have more than one table? at some point you need to get info back to EP - it is elegant to say only last query gets to return results - if you have one, fine but what if you have multiples?
- JB - merit in being to accomodate more than one table - argument keeping separate makes it easier to adopt openIDL in the future is a good one
- JM - 2 issues: a. schema separation (org govertnance and security concerns) - day 1 EP submits whole script and you get one table back - there is an interesting question in how to get it back
- KS - DR felt you don't need sep schema
- JM - technically DR is right, design constraint: big orgs nutty about db power, 2 schemas solves org problem, 1/2 dozen operations, set of scary rights DBs worry about results in log discussion

- KS - diff argument, project: make sure we can do something, maybe not MVP but run a single script that can do the stuff we need to do in a single script in HDS and it does work, now lets figure out will it pass muster in policy groups then go to second schema to solve policy issues
- JM - policy question not an empirical question - are DBA teams nutty about power
- KS - for H this is important, for T it didn't seem to be the same level of importance
- PA - for AAIS feels weird to give that level of power to raw tables
- JM - if we have good modular design should be fairly easy to add loops to the base (scripts), design day 2
- JM - ask we dev in highly modular fashion, compnentns have no external coupling, constraining ourselves by removing loop - you have 000s of queries in script, 2 classes
    - 1 does something
    - 1 returns results
    - do there need to be two scripts?
- KS - into the weeds how you return results? populating table?
- JM - n number of queries and commands, lets assume EP in javascript you have DB connection/objects (run these commands), hey engine pls run this query - dml command, return code, run script that can return command, have populated memory struct in javascript, "execute query, return results"
- KS - script or sql command - running a sql script running commands, still able to return results out of script
- JM - dont think we should run a script - "what client am I pusing this through" - resp of EP to scan and submit each query to postgres engine
- KS - differing from yesterday - how do we get results from script? prob can't - jm doesn't want to run script
- JM - challenge is getting results back, run script is easy but if we do needs andother step, run script need to run a client, easy to tell client run script but challenge is getting results back
- PA - if i am runnign script as part of EP, some ID associated, can't i make a table in results schema I loaded up
- KS - need schema to see final table
- JM - no you don't, if i submit EP and has 10 SQLs in it, and tell EP "run scriopt and loop through", runs 1-5, last query returns result set (might create tables, views, drop tables), EP needs some mech to identify which is the query to return result set - some set of commands to set up, set up table and breakdown scripts - design of object has any number of setup commands, singular query command (returns result set) and any number of teardown commands

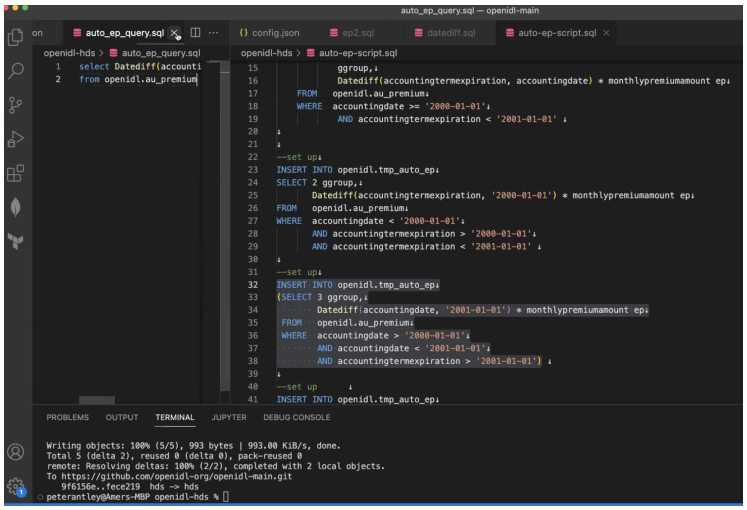


- KS - back to a collection of commands
- MZ - just using sql commands? track versions?
- JM - currently just commands
- MZ - could have versioning for data definition changes
- KS - initial request might migrate over time
- JM - notion of data versioning is true, HDS has versioning, how do we pass script in, "queries compatible with x", design of scripts/queries is versioning, need ot make sure HDS is aware of version and EP is aware of version and can be compatible
- KS - one case of using collection of commands is just using, run them against postgres engine and last will be the result (one in the middle will be the result, one identified
- JM - mult query patterns at the same time... could get rid of looping entirely if we allow for query commands that return result sets if we interrogate them - any query that starts with "SELECT" returns a result yet - EP loops through all queries in script, hits "select" query
- KS - how do we pass the results of one command to return results
- JM - script writer has to do it, 3 setup commands (create tables, etc.) the EP runs each query doesn't know doesn't care - query writer has to know
- KS - how do you get results out
- JM - EP establishes session, run it, EP writers job to put commands
- PA - will deliver setup and result - three scripts: setup, result
- JM - getting rid of loop, any number of commands can return result sets, EPs obligation to interrogate those queries - if it starts with SELECT returns result - still loop in there thru the commands but super fine grained the loop, instead of looping at script level loop at the command level -
- JM - will look like a script, submit it to a client that submits it to the db, we are becoming the client, we need to loop, no magic, make basic rules (semicolon, blank lines, simple parsing)
- JB - comments in the script
- JM - do we put directives in?
- KS - directives nose under tent
- JB - fragments telling whats in next set
- JM - no command returns result set w/o being called SELECT
- JM - questions of pagination, etc. - if we agree we loop thru every query, when select we result set, MV1 dont return more than 10k rows
- PA - none of the reports we do early will have more than 10k rows
- JM - will have to deal with memory and pagination issues after Day 1 - hide attribution and fine grained info - intermediate should never leave db engine
- KS - if assuming says "select" and captures results - instead of allowing mult returns only last select statement?
- JM - day 1 = one table
- PA - all standard ones will be one table (there are non standards)

- JM - hard part of mult result sets, whats hard is 2-3-4 results sets how do you return via API, maybe we say API returns 3 huge chunks of JSON and the receiver figures it out
- KS - still stuck on SELECT returns result and others
- JM - only select statement returns results
- PA - for all the annual DOI stuff doing, in good position, need to talk to person at AAIS, lot of low hanging fruit with design
- JM - if only have select statement, MVP 1 run one select statement
- JB - flexibility from diff lines of business, combine and consolidate
- PA - do it all with intermediate temp tables
- JM - EOD 100 lines of JS, suprisingly tight, harder to understand than code
- KS - simple in JS,
- PA - other option, Extract would have name to it and fill table based on name, tear table down - result set in persisiten table noted by Extraction
- DH - perm table stored where?
- PA - your HDS
- JM - or the EP schema -
- DH - another thing to consider (maybe day 2) - companies will want to see results of queries run on their data
- JM - had that accounted for (test/eval or scanner validator)
- 

Mon., Oct. 24, 2022

- KS - diagram shared last week - depiction of how work should flow. Some requirements were embedded in what was said. (Document shared illustrating this).
- KS - high-level: bringing a script in that is parsed into commands. Starts by querying HDS, can do intermediate tables for holding successive queries, can return results on any of said queries. We can define queries as select statements. Parsing into collection of commands and executing those commands vis-a-vis 1-2 schemas. We will be walking through this today.
    - KS Queries only against HDS - can we assume the first query we see goes against HDS and no others will?
    - JM - no, but this isn't a problem - it's an asset
    - KS - We have to say which schema we're querying. JM: correct, but these should be reserved word calibre notions. The schema names are extremely well defined.
    - JM - we should prefix them with unusual enough names that it will forestall an issue, prescribed in a hard-coded manner, so that they can be used again and again. We should be highly prescriptive. **Decision: openidl_base, openidl_ep.** We will require that the select identifies the schema. JM: another aspect: the application user running the query can only select from the base, much more authority in the EP schema. KS: in fact, user will have select authority against both, but only more advanced rights against EP.
    - KS: Only really one of these that is required - the initial select. Everything else is optional.
    - KS: how do we draw in sequence diagram? JM: open session doesn't actually go to schema - it's at the PostgreSQL engine running so it can be taken out. Sufficiently accurate to say you're creating tables in EP schema, such that you may select against the base. When we get to the level of detail where any more depth doesn't give us add'l insights into requirements, we can stop there.
    - KS: will take tweaking offline. (PA asked KS to pull link to Github from chat, and PA shared screen)
- PA: Everything is in the path, so the premium amount is the actual amount.
- PA: in a really perfect situation, this is all we need to produced earned premium:



- PA: Earned premium is not really about one's data set, but about EP within a certain timebound. To get this, we take earned premium/months covered. This is very basic, and data set will never be one like this.
- PA: Working with SQL - hard to utilize parameters without an extremely advanced client. I have start dates and end dates, and strings, but they are parameterized when actually used.
- PA: We begin by running a setup command and creating a temp table, called Temp Auto & Earned Premium. Begin calculating group 1, group 2, group 3, only 1 creates the table, next ones insert into that table.
- Extraction for EP comes down to selecting sum from temp table. Teardown at end is drop of 1 table. 8 columns - EP is first column. Second part - not only earned premium but EP by coverage - column by column. So breakdown will expand significantly over time.
- KS: current path is sufficiently robust to implement this. PA: this is actually a single select statement (KS pointed out not a single select to implement the whole thing).
- KS: Line 14 - this is a create table statement not a select statement. We have to recognize HDS schema request as a select statement.
- PA: What we need to do is take these commands, break them down, and separate parse them into objects - e into queries with semicolons.
- KS: This will not work - won't come up as a select statement. We therefore need to break it up into script, yes?
- JM: No, will be fine -after tokenizing this each token represents a query, within that larger query. Show the first token within those larger tokens - if it's select, it will do something, but it doesn't matter, because it's the responsibility of the query writer to identify the two schemas in question.
- JM: All scripts assume their default schema is going to be openIDL_ep - we only specify schema for occasions that involve reaching out to base schema.
- PA: But when we're doing the create statement for EP... we should go openIDL_ep?
- JM We leave it blank. Because it is so common to reference EP schema
- PA: I'm talking more when I *create* the database.
- JM: but in creating base schemas we want to specify schema name, absolutely. (KS: this happens outside of extraction execution. PA: but pivotal to do this in setting up HDS).
- KS: The default schema then is the EP schema, therefore it doesn't have to be mentioned anywhere even though it can be but doesn't have to be.
- PA: It may be that we start to have some persistent more advanced tables in EP - recurring structures.
- JM: This proves our point and this is exactly what this code tends to look like in terms of business logic. From a pattern perspective, there is different logic for 1 2 3 4, correct?
- PA: Correct. We process records differently depending on when they fall in the year vs. where they are extracted.
- JM: This is a source of confusion - i.e., what is the difference between line 25 and line 34. (He got clarity on this).
- JM: This demonstrates the case of being able to make intermediate tables. For people who have to read it comprehensibility is a must.
- PA: For EP - could make more sense to utilize a function and build it differently.
- JM: For scalability & modularity, the desire for functions explains the need for EP schema separation from the HDS/base schema. It points to repeat question of should this be in the extraction patterns or DB definition?
- PA: For auto, with earned premium, if we have this, and the corresponding function exists in the architecture, we can have a good test set and a high level of confidence that the function works correctly.
- JM: Business rule management is critical here. One of the requirements from business people: they have to be able to manage these rules and comprehend them in a useable way. If we can start getting acute granularity - 1:1 on requirements and functions - there should be a single functional entry point for every designated business rule. This is incredibly powerful re: the scalability of human comprehension. Functions should be defined in either the base schema or EP schema but not buried into extraction patterns. They are too stable across time.
- PA: We're going down an interesting path that will put us in the realm of metadata-driven architectures or table-driven architectures.
- JM: we can put business logic into code modules, aligned to business requirements in written form, we will see repetition even within functions that need to be tabularized. This will put us in a highly scalable architecture/part of a scalable solution. We can talk later about how much we will scale them.
- JM: this is critical if we ask Mr. Antley and others to start coding - a key differentiation.
- JM: We also ask selves if functions should be built into base data? It's not uncommon to venture down a path where we ask if we can do this with ETL time.
- PA: we can't do this w/ETL time - until we know what our bounds are, we can't calculate the results.
- JM: We should assume we want to use functions at this point.
- PA: a ltd # of functions should be part of bedrock infrastructure from outset, but not created at query time, otherwise SQL query will get so large that most people can't even read it.
- PA: With the power of functions, I can do one select statement and get results comparable to auto coverage report. Functions we're talking about right now will be created before extraction is run, + validated and tested. So we aren't creating functions at run time. They will be built into the database and utilized.
- KS: Disagreed strongly with this, because we have to govern those functions and install them. Doesn't believe this is the right approach - we want to keep it as simple as possible and just run the script. Doesn't see the value of creating these elements ahead of time if we can create them as part of the script.

- PA: we have to do governance regardless, and not many functions made. e.g., 10 functions, a test bank to prove they are correct - govern these far more easily than, say, giving someone the ability to create the function at runtime. The amount of validate done w/extraction patterns goes up exponentially without the use functions.
- JM: the idea that we're codifying a certain body of knowledge in a schema is fairly standard. The logic has to exist somewhere - question is not does it need to exist, but where do we put it. Schema definition - putting it in the schema build - is a much more stable location. The question: does the pace of change match the pace of the place where we put it. Pace of extraction patterns - extremely high and dynamic and different every time.
- KS: not challenging logic but timing. Doesn't believe we need to go there for the MVP. More complicated parsing.
- PA: the amount of renaming (reformatting semicolons, etc.) is overwhelming. But we would set a delimiter.
- PA: we are likely taking about 9 functions. First is DateDiv (takes 2 iso dates - givess us difference btwn dates/months). 1 function for each column in that table.
- JM: If we can develop a set of functions that map cleanly to business requirements - lucid enough that any businessperson can read it - this represents a huge strength.
- KS: we've added a significant dependency on strong governance. If the code is in the install base, the latest version at any time will be necessary for a user to run extraction patterns. Key consideration. (Potential drawback) We just put a huge burden on everyone having the latest version of these helper functions. Very concerned about the governance here - ensuring functions get into nodes, and making sure everyone's nodes are updated.
- JM: but no worse than the fact that we'll need to govern it anyway, and there will be versioning regardless. We will have the same challenges regardless. Without functions, all logic has to fit inside extraction patterns. Then the database is humble, we have to add more and more to script w/additional calculations, we could end up with massive extraction patterns. Massive pre-work to get one basic result. Ignores our ability to take advantage of the inevitable repetition within our codes.
- JB: We will have governance-related obligations on the database anyway. Changes will be less frequent than we fear - may only be once or twice a year. Stability is key. JB wants to try to put a higher-level overview on our discussion -re: impact of the adapter wrt its functionality and the operations it might need to perform.

Mr. Braswell presented a diagram w/various logical endpoints of an interface for API and the various types of resources that might be used. One of the central ideas behind this: this can and should work whether these are call-ins or call-outs. The diagram illustrates that there are other types of interactions in the adapter to the hosted node in addition to just processing the data extract:

- Details of how we perform the extract can be summarized by the two ellipses at the bottom and the middle <initiate data call processing> and <return data call results>.
- The diagram illustrates something fundamental: Since we are positing the notion of an interface to a hosted node, some of the functions that exist in the node itself - re: looking at the queue of requests or the UI that interacts with one of the Kubernetes containers to talk to the chain code - this particular function (knowing what the requests are/what's in the queue) needs to be proxied out into some form of API call in this interface. JB is calling this "data call management" - i.e., the notion of refreshing data call queue, is the equivalent of getting the current list of data calls that might be seen in the UI on the actual node, but we aren't assuming that someone has this UI because they aren't running that node, and they lack the container in the Kubernetes cluster. We could actually call out and say "please let me know what all the outstanding data calls are," which will yield a list - and if we want to look at a particular one, that view data call request may give us the ability to look into the detail of that or make a call out to say 'let me know what this request is.'
- Like and consent are notions that precede actual notions of the extract. JB has put the extract processing under data call processing and we start this with 'get data call request.' Whether or not this is something that is called in or called out, there is then the notion of approving data call request - part of the validation process. Making sure we want to run it. This doesn't have to be an API call, but we should have the option not to run it. We can also at least acknowledge it and say to the other side "I'm beginning to run it."
- Returning the results completes this process. The final column, Data Call Disposition, has a notion of Data Call Results received - did you get it? (Confirmation of receipt)
- Two-phased consent - after is all is said and done, is it okay to release the results to this carrier? Total aggregated results etc. Called 'release and final consent.'
- Results of disposition of data call - last step - what is to be done with resultant data? Is it deleted? Is it stored? Is it archived? (i.e., Closure)
- Overall, this diagram shows, logically/conceptually, what some of the major interactions might be in the course of the life cycle/data calls, that would abstract this directly through a hosted node UI, without the use of the UI directly into the chain code. There does need to be some type of agent or endpoints that will service this interface regardless of whether it is pulled or pushed.

Discussion Ensued

KS: Raised question of what is in hosted node vs what is in carrier's world. Proposed that we may have great difficulty installing all this on a carrier.

JM: The reason on the data call management side, that the refresh data call queue exists, as a message, is that x user doesn't have access to the Kubernetes running on the node that queries the chain code to get the outstanding request... So all of this has to do with messaging that is talking to the hosted node via the interface. The adapter says what are the outstanding requests, sends back a list of current requests - a roster - in message form. Kubernetes would be on the other side of the hosted node interface. We can do this with a very simple script that looks at a table or a file?

JM: Questions. We say "hosted node" but what is the official logical name of the other node?

JB: This is the side of the interface that would be the adapter, that would know what to do, to talk to the hosted node.

JM: There is a box we call the carrier node - and it has an entry point/API with various functions. But what is the name of the thing calling this?

JB: If one has one's own node, one doesn't necessarily need the hosted node interface. One can perform these tasks directly. But for the purpose of the carriers that don't want to host their own node, the concept of the hosted node - this is the fabric node that runs on the network, but the fabric talks to it, so it isn't necessary to maintain that in one's own carrier space.

JB: all of these mechanisms sit in the adapter on the carrier side.

KS: We need to refactor the naming of this - the blue box that makes it work, that's in the carrier - more than just the adapter, it may be necessary to deploy some of these other pieces.

JM: What entity calls the API? Who is the client of this?

KS: As we're doing an extraction for a data call, we're interacting with this part x to say "I consent.' Upon consent, the chain code says "I need to get data from the carrier." There is a trigger - something's listening in a pod in Kubernetes (a trigger). This is an application called the extraction processor that lives in Kubernetes in the hosted node, right next to the chain code (JB: a node on the network - has functions that interact with HL fabric). JB: consent has to come from carrier, can't come from hosted node.

KS: Yes it can, because the carrier's interacting with the hosted node... There is a UI, we have to use the UI running on the hosted node to interact with the data call. That's what this diagram depicts.

JB: I'm assuming that it isn't necessarily going be that easier for a carrier who doesn't have the node to access the UI that is running in the node - this may present security issue for getting access to the node?

KS: This can definitely present security issues, it's just a question of getting permissions. Basically it's SAAS at this point, something like an Atlassian Confluence. Just interacting with the system, it's hosted elsewhere. That's what this picture says - something running over there has a UI that we interact with, via permissions from the host. At some point it reaches out, because it needs to run an extraction, sends the extraction across, and that is when this takes place.

JB: We shouldn't assume implementations of full function node, in cases where you're running your own node with the UI and the different containers that make up that cluster, if there's a dependency on getting access to this, the interface btwn a carrier and the hosted node as a svc could be done with this interface, so that you don't have any dependency on the implementation of the hosted node but you do have an interface to talk to. Ergo, whether it's a call out or call in, it can be terms of not necessarily having access to components inside the hosted noded self.

KS: It's not receiving from the UI, it's receiving from a module running in the hosted node - the extraction processor program, that grabs the data call, takes the extraction pattern out, sends to the adapter, and says 'Please run this, I will take the results and put the results on this channel so it goes through the analytics node." The results are then transformed into a report by analytics node. We're splitting out the execution of that extraction pattern into the carrier world, because that's a stack we think will run in the carrier.

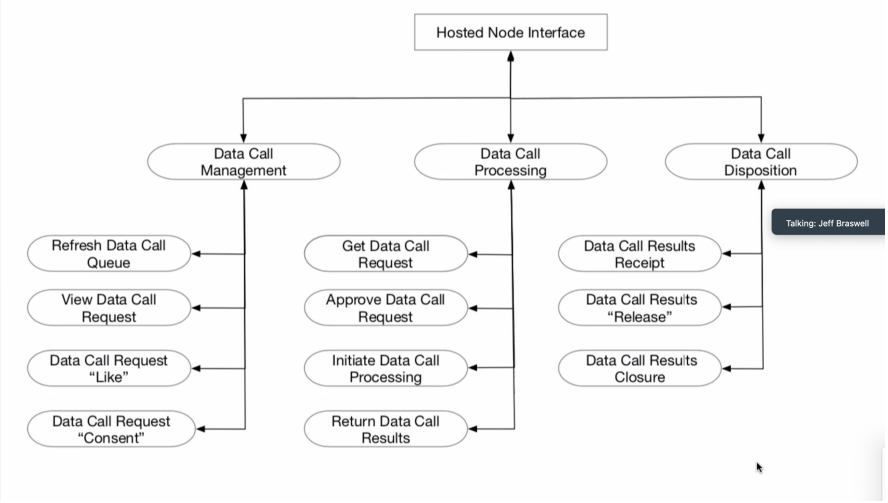
PA: Hosted node will send a request for data to the member's enterprise, member's enterprise returns the data back to the hosted nodes, and the operator will utilize UI coming from the hosted node. No operator will be operating on the enterprise.

KS: Actually if the enterprise wants to run the whole stack, this should be an option. We've laid out the above because we want to get a tech stack that is approachable and one that includes HL fabric isn't approachable in the short term.

JM: OpenIDL as a service is not a hosted node - but an environment in which you can host any number of member nodes.

JB: But there would be a hosted node for each carrier.

that if there's a dependency on getting access to this...



Mon., Oct. 24, 2022

Recap

KS: Last meeting - drawing boxes and labeling what is what - i.e., what is openIDL, non-openIDL

(Mr. Sayers presented this diagram again)

KS: We reviewed adapter piece - aka openIDL Data Module - this is the piece provided by openIDL to run in carrier's cloud, and we also have hosted control module that could be in cloud also but is separable (per the Travelers approach). Some concerns about what information/data goes across this line on the open internet. The security needs of this will be revisited.

KS: We also talked about the extraction processor itself, and executing that script - there may be some functions as part of this. We've discussed about running a script that establishes those functions first - these scripts will use these functions repeatedly, we build them before query time. Parsing functions in - will change the delimiter and make it much more complicated. Executing as a set up script means alleviating the need to parse it every single time.

JM: We can either run it as a prescript or build it into database structure itself. (KS agreed). Pros and cons to each approach. Part of extraction process review - reading and signing off. Increases demands of individual on front end. If we had both we would have far more flexibility but incredibly complex engineering.

DH: Biggest concern - what is going out the door, and how have those attributes been created.

JM: Any function we define should have business requirements from which derived, and a unique identifier - then when a coder comes in there is greater clarity. Code has to do what requirements say it should do. By having it in the DB layer, we test it assiduously. Traceability with code + stability. The opposite problem if we put these functions into the extraction pattern.

JM: In final design, one of rules has to be... (clarify)

KS: do we have a sandbox that we can define and protect around these formulas?

PA: Shared screen - and formulas for EP, Car Years, Incurred Count, Incurred Loss. All these calculated based on spreadsheet Mr. Harris made. PA went through query that he uses to make car years with various included functions (car years, earned premium, incurred count, etc.)

KS: Can we constraint the go to the EP schema?

PA: Yes, depending on who makes the stored procedure, but we will absolutely need manual review. We can get very explicit about which permissions we give to which users. Question: do we want to just be doing an extract to the EP so function can be work on that instead?

JM: Functions in EP schema make more sense

PA: Do we want to replicate the auto premium table from the base schema into the EP schema? JM: No.

PA: We get the procedure which exists in openIDL ep schema - running a select on a base table. Is this a problem?

JM: No, and this is why we want functions, procedures, etc. in ep schema - so we can restrict hDS functions. It's a safeguard.

PA: We will not replicate the base tables, then.

PA: Running through table - it begins to break up information. Still looking at how to break out physical damage into comprehensive & collison.

KS: We have the ability in a best case to get functions into code - present and available as part of EP schema. Can we say all functions we be only against ep schema?

JM: They will not have manipulative authority against the HDS. There is a safeguard even if we put functions in the call. We will have hundreds of functions over the years. We want to allow for a few novel functions in the calls but don't want to stuff everything into extraction patterns. We should provide room for both.

KS: Where is constraint on HDS manipulations?

JM: We would grant select to EP schema, but wouldn't grant anything else. The EP data only has read access to the HDS.

KS: We execute prescript as a script, the parse extraction script into commands and run the commands. Can we put another delimiter in there that delimits everything?

PA: Question - currently has a premium table, a loss table and various functions. can write a bash script, -e., etc., but how should we structure deployment?

JM: Many small scripts with one orchestrator across it (orchestrator can be a bash script).

KS: Reservations about bash given about its openness.

JM: At the low level of hitting a database with files, there isn't the need for an interpreter associated with Javascript.

PA: will develop bash script for this and also work on breaking out all rows. Will run multiple psql commands (terminal commands for accessing database), create databases, create schemas. Script will be very carefully regulated - this will be core database structure. Very carefully & deliberately infrastructured.

PA: Provisioning the server will mean doing git clone.

JM: In looking at bootstrapping, we need to say do we have a Linux server to work on? Then create user, create schema, etc. This is bootstrapping. We need to determine if we want to pre-containerize.

JB: Setup/configuration before we even perform first extraction.

KS: Extraction pattern is not bash script. Just SQL.

JM: We have to build module on the carrier before hand. What is the extraction pattern allowed to assume is already in existence? Set of well-defined functions.

How does extraction pattern look?

KS: Two text docs - one creates functions that aren't there yet, and one that can parse individual commands; each one executed in turn. Extraction is still in one file - command separated by semicolons.

KS: Asked how we're executing extraction?

PA: A service user will be installing database... A javascript wrapper will connect with credentials for service user. Giant fixed width data array will come back. Java script - executing this against database.

PA: The one element we don't care about - definitely connecting not with local host to lambda function.

PA: Spike POC needed to run a javascript code against the database - PA is happy to develop this.

JM: Node_JS prompts interesting discussion regarding AWS Services - quite a big question. Will I build with server lists or not?

JB: Some carriers may not use containers on their sides.

JM: Containers however tend to be more universal. If we build this for Linux, and if we're comfortable doing so, we can build in both containerized and native versions.

KS: Difficulty will arise in connecting to API

PA: Two options API gateway or Javascript Express.

JM: We should lean more heavily toward Javascript - basic linux.

KS: begin with reference implementation using AWS. As long as what we're replacing in ref. implementation is obvious, it should be fine.

JM: Suggestion to PA - build postgreSQL database on Linux server. (go back in recording for elaboration).

PA: In the past, he has used Python Flask to do API, and manipulate exposing the port/doing the framework.

JM: Node_js libraries should include the framework.

PA: Would probably want to use express.

JM: ???? Send and receive on the same port

JM: If we have 30 parameters and all are defined, it's exposed. We'll get e.g., 25 returned values. Humble, more flexible version - any characters should be capable of being validated. Caller needs to know how to parse Json.

PA: In payload, JM wants to put json in return value

JM: Or we tell it that it has multiple paths. One Json string in, one big string out. Json validation necessary. Is this okay? String out is result set, we need to agree on how to jsonize this etc. We also have to decide how to paginate. We need to define page - API calls are not meant to hit a browser. We will hit against a result set that is ridiculously large

KS: Going across carriers and combining at end - different zip codes, etc. - we may end up with massive results.

JB: At some point we need flow control through pagination

JM: Recommends that we table it - lg strings in and out. Json schema at least for input. Prescribed format.

JB: For output, if we get into doing different type of report we will say x for header...

Time	Item	Who	Notes