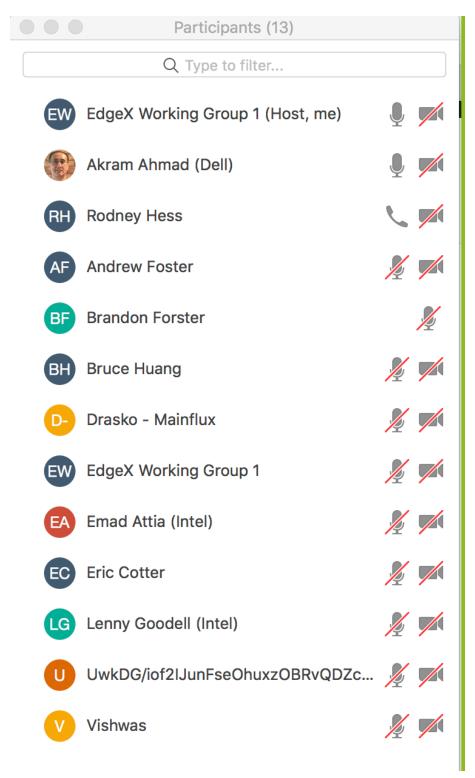
Core Working Group Agenda (01-Nov-2018)

Attendees:



Old Business

• It's all new business, baby!

New Business

- Developers responsible for blackbox-testing changes
 - o Integration test ownership
 - Indicated by
 - Changes to the API signatures, reflected in the RAML
 - Non-equivalent functional change to existing endpoint
 - General agreement from attendees
 - Jeremy raised CI concerns
 - PR → verify job → If pass, merge & test is triggered. If test fails, then issue is created for PR creator.
 - Can integration run as part of the PR pipeline and merge not occur until test passes?
 - Can devs run blackbox tests locally before creating a PR?
 - Documentation
 - Blackbox PR → docs folder containing the Postman/Newman PDFs (Trevor)
 - Record a walkthrough of local blackbox testing procedure (Trevor / Andy)
 - O Question: should we consolidate code/test repos?
 - General agreement, will be easier to coordinate and ensure feature complete.
 - If scope indicates need for test changes, inclusion in code repo will make it obvious whether tests were changed or not.
 - Test coupling to Mongo
 - Discussed rework of tests to reduce reliance on pre-seeded data that might have coupling to particular platform.
 - Proposal to have the test script auto-gen IDs where necessary, perhaps based on an environment variable setting.
 - Discussed IDs versus Keys as part of data seeding for tests
 Should IDs even be exposed.
- Structured Logging
 - GoKit Logger
 - DEMO
 - No objections on the call.
- Go modules
 - First steps
 - Devs upgrade to Go v1.11.1

- CI jobs updated to use same
 - Dockerfiles need to use new base image
 - What about SNAPs?
 - Single line change in Snapcraft.yaml file
 - This is being discussed in the weekly DevOps calls
- Repos can now live outside of the GOPATH
 - Repos within GOPATH that have been converted to modules must be compiled used GO111MODULE=on
 - Should be easily accomplished in both

WILL COVER ITEMS BELOW NEXT WEEK. RAN OUT OF TIME.

- Database Abstraction, remove Mongo types
 - o Discuss exposure of db generated keys outside of service boundary
 - Keys versus UUID in usage
 - Key, represented as string value is DB specific, cast by provider
 - Mongo compatibility
 - UUID is a standardized unique identifier
 - Could potentially be supplied by upstream application, not database.
 - o Go libraries for UUID generation
 - https://github.com/google/uuid
 - https://github.com/satori/go.uuid
 - https://github.com/nu7hatch/gouuid (OLD)

Supplemental notes

- DB → Business Layer ID representation
 - Answer question → UUID equivalency w/BSON ID
 - If not equivalent
 - Key (string type) DB PK value
 - UUID →std library type
 - Modify query paths accordingly
 - Indexing in Mongo for lookups

UniqueIdentifier (GUID) **

Key (BSON) - DB Key ** may not want this exposed **

Both will be strings as raw data types

```
"created":
1464039917100,

"modified": 1474774741088,

"origin": 1471806386919,

"pushed": 1471806399999,

"device": "powerScoutMeter",

"uuid": "my_uuid"
```