We have to get a consistent ETL framework running, comprising the following parts:

A scheduler / queuing system with the following characteristics:

* A queue table that be inserted into by external systems
* An inserter to put loads into the queuing table based on schedules
  + EG hourly schedules
  + Daily vs monthly
  + NB: The schedule table should reside in a separate DB from the queue and loads tables
* A Load table that contains what packages make up a load
  + Step order
  + Max Parallel : e.g. we could have 10 dimension tables all in step 3, which are then run in parallel. But if we have a 4 core system, we can set the max parallel to 8, and it would run 8 in parallel, then 2.  
    See the Parallel ETL package I've put in GitHub
  + NB: This doesn't mean we throw away SSIS control flow  
    We would still have some controller packages, and this allows the developer to choose to do some items through the load, and some through a controller.
  + For example:
    - Load first does Business dimension by calling a controller
      * This controller runs 3 packages to load the complex business dimension
    - Loads contains all remaining dimensions in parallel
    - Load contains facts with no dependencies, runs them in parallel
    - Load also contains a FactInvoice Controller, which calls FactInvoice, then FactInvoiceLineItem
* Load type flags
  + Prerun: Information\_schema compare
  + Postrun: Balancing tests
* A logging DB:
  + In a separate DB from the load and queue DBs
  + In case of ETLs where no logging has been configured, eg Zero1, the queuing package contains logging for start time, end time, success for each package. For clients such as Zero1 and Genasys, this connection will in fact point to one of our Azure DBs
  + An SSIS job that scans the logging DB for failures and sends an email will then run on our one of our Azure servers
  + This job will also check the schedule table against the logging table, and if an ETL hasn't started according to the schedule (within a time limit set in the schedule table), it will send out an email saying this hasn't started. This insulates us against the ETL being completely down as this is a separate process

Prerun:

* Information\_Schema compare runs an ETL that retrieves Information\_schema data, and stores it in a table in our configuration database, including a run ID
* The package then does a comparison between this run and the last (of the same type), and inserts a notification into the logging DB if they differ, on either the source or the destination
* This allows for much quicker debugging if a process fails
* The prerun flag determines if this is run - eg hourly loads it wont be

Balancing:

* A set of predefined tests are run against source, the destination DB, the cube, and compared against each other and for some tests against a value stored
* Mismatches are then mailed out

**Items we need to build:**

* Databases. All databases go into a single DB solution, but a project for each.
  + Scheduler DB : Geoff can you prepare a briefing on this?
  + Queuing and load DB: Geoff, again you: look at the requirements above vs what you have and look at what is missing
  + Prerun structure: Christina, you're the most familiar with this, we'll need a new structure
  + Logging DB: Some parts tables for SSIS logging, but we do have multiple clients
    - We could do a multi-tenant?
    - Or have a logging DB running per client on Azure
    - Geoff, can you reuse anything from the RMB design?
  + Balancing DB: Grigori, can you prepare a briefing on this, and look to any outstanding items we need
* The SSIS code
  + A lot exists already in GitHub, we'll use this as a base once we have the DB final

I suggest we all prepare, and do a meeting on Friday. This is a more complex controller set than most implementations, but will give us a much more fine-grained control over the process as well as a backup for no notificatiosn, and automating finding some classes of problems.

We'll also add other items later to the pre-run: eg disk space warnings, As these are mostly covered by other tools, this is much less urgent.

As this is a cross-cutting project (I.e. for multiple clients), Christina will be leading the team. I will be involved early on and then step back:

In terms of your current project commitments: If you are fully booked out at a client, I'd like to pull at least 4 hours a week off into this: (Matt only once your Dax is finished, so you're on the bench): Grigori, this will qualify under Zero1 time.

Geoff, do you need me to talk to anyone at RMB?

In case any of you would like to work on this  overtime, I'm happy to allocate up to 8 hours a week over and above client work to qualify for billability bonuses - as always, this is at your discretion. Bear in mind that this is a fairly strategic internal project, and the code you deliver will immediately go into production on 3 projects: Zero1 OneView, Zero1 OMS2, and Genasys, so we are attaching importance and urgency to it.