# Response to Answer Dev team suggestions.

Ian McNicoll

"As requested see below a list of the things I can think of that which we would need to be done on the Ripple project if we are aiming to make something which can go live.

1. The Create Composition Routes need to be fixed for, Allergies, Contacts, Medication and Problems.

This will mainly include changing the check so it points at the find composition route rather than one which does not exist. If an existing composition is not found it would then create one. If a composition is found then the camel should redirect to the update composition route so that the new details can be added to the existing composition as a new object in the repeating group.

This would probably be a good time to check what should be included in its own composition and what should be an object within a repeating group contained in the composition. I think this was developed under an assumption when the system was first developed, so would be worth querying before making any significant changes.

Currently all the types of information, such as allergies or medications, are stored in a single composition with eath instance being an object in the repeating group or the composition. The only exception to this is the MDT's which all have their own composition, one MDT to one composition.

**IAN:**

**I think there is probably a little bit of confusion (understandable!) about the ‘commit strategy’ i.e where and when to use POST vs PUT.**

**Composition Commit Styles**

**Depending on the clinical requirement, 3 styles of commit strategy are suggested.**

**‘\*\*Event’\*\***

**Typically this is used where each commit of the clinical document requires a new instance to be created and those instances are generally all readily accessible. Where an existing instance needs to be updated this is generally because of an error.**

**Tech: Each time the composition is committed, create a new instance via a POST. Typically for Nursing Obs, clinical noting, probably MDT meeting reports.**

**‘\*\*Episodic’\*\***

**Typically this will be used for a clinical document which is maintained as a single source of truth for the duration of an period/episode of care i.e routine querying should only find a single instance of the document. We expect to routinely update the same instance as part of routine care.**

**Tech:**

**Create a new composition via POST for each Period of Care i.e an admission. If it needs to updated use a PUT to modify i.e Each patient has a single instance per Period of Care.**

**‘\*\*Longitudinal’\*\***

**Typically this will be used for a clinical document which is maintained as a single source of truth for the patient’s lifetime i.e routine querying should only find a single instance of the document. We expect to routinely update the same instance as part of routine care. Examples FP problem list, allergies list.**

**Tech:**

**Create a new composition via POST for each patient. If it needs to updated use a PUT to modify i.e. each patient has only a single instance over their lifetime. This will be unusual in a hospital record where there is generally limited ability to curate the patient record in this way.**

**In the Ripple use case the Allergies, Problems, Contacts and Med lists are also ‘Episodic’ (arguably longitudinal). End of Life Care is very clearly longitudinal**

In other projects we have been using a Cluster archetype in the Composition other\_context header to capture episode information. We could also usefully add IHE-XDS metadata and other document classification which would make the compositions play nicely with XDS and other no structured documents like CDA level 1

2. The Models should be updating to include all the mandatory fields with appropriate defaults where the gui does not ask the question. Defaults are required as the record will not save to openEHR without all the mandatory fields.

**Ian: Agree. You should be getting defaults for mandatory fields (where appropriate) from the templates. It is possible to ‘bail-out’ of mandarin by using null\_flavours but this needs very careful clinical safety assessment.**

It may also be worth storing the raw composition in the JSON like we do with the the MDT so that when the composition is re-written we do not loose any data which is not held in the model. This may happen if another system creates the record with additional or new fields then our system updates the records without knowing anything about those fields as they are not in the model. The other option would be to re-read the old composition when processing the update request and overwrite only the information we get back from the GUI and leaving all the other fields as they are.

**IAN: I suspect the commit strategy for MDT is incorrect. It sounds like you are doing this as either longitudinal or episodic, when I think this is probably an Event strategy use-case. It is a but subtle and when I spoke to the UCH clinicians I did initially think this would be best as an episodic commit.**

**If you do use an Episodic strategy the correct approach is to read in the whole previous composition and update it as a whole. This might mean appending some new its but equally it might mean deleting or changing existing items.**

**This is the sort of approach that we definitely need for something like and End of Life Care document which is continually updated as a single source of truth. Int retrospect, and with understanding the use case better, I don't think this applies to MDT reports. I think the composition handling will be easier if ww area these as single events.**

3. They may want to generalise the Cancer MDT, if they wish it to be used for different MDT types not just cancer ones. The naming conventions and routes would have to be changed. This would depend on the how we would get the data from openEHR as we would probably have to update the aql query and the model to hold the required data. It may be worth asking if they are looking to expand the MDT menu to include a selection of different MDT's not just cancer ones.

**IAN: Rather than generalising to different MDT types, I would suggest exploring the requirements of the Prostate Cancer community in more detail. Whilst a nice tech demonstration, the current MDT does not meet the needs of any of the key clinicians involved. I think you are going to find that while there is some degree of commonality across various cancers and non-cancer diseases, which we can reflect in the underlying archetypes, in practice we will end up with very different templates and associated UI.**

4. The method we are using in the the MDT Find and Update code, where we make multiple calls to openEHR to get all the MDT compositions and then aggregating the results may be better designed, so it would be worth looking into re-designing and improving the code.

**IAN: See above - I think a change of commit strategy on MDT will make this much easier.**

5. All the routes need a better/consistent method of storing information when making multiple calls to the openEHR service as currently I have added a simple bodge to make it work, and to stop an error. The solution is fine but it would require tidying up before it could go live.

**IAN: Need to understand this better.**

6. The system is currently a bit slow and could be improved if we were to change the order in which the code processes information and when calls are made to openEHR.

To fully tackle the performance issue with the system, especially for the MDT route it may be worth changing what information we get from the GUI, as currently we do not know which MDT is being updated so we have to process them all or we could possibly do checks before calling openEHR to update a composition as this is where the bottle neck is.

**IAN: We can add metadata to the composition header and templates name to make it clearer which MDT team is involved and speed up querying substantially.**

7. There is some inconsistency between the PUT and POSTS, only really for Allergies which it may be worth changing if we can.

**IAN: This should be simple. The first creation is always a POST, subsequent calls should retrieve the latest composition (should only ever be one) and use this as the basis for the next update which is a PUT.**

8. Very importantly we need to put some sort of error handling in to feed useful information back to the user as currently if something goes wrong such as someone updates a record with a field containing an invalid value we get a 400 error response from openEHR then just pass that back to the GUI which does not know how to handle that and the user does not have any indication why the update or create they have done has failed."

**IAN: Agreed - there is a more detailed error message carried within the response but this is of a technical nature, really for debugging and ultimate  and not appropriate for user feedback. I have checked with Marand and they do not attempt to do anything clever with the error message but rely on validation checking in the UI. As far as possible they generate the validation automatically from templates.**