The Topic

- LINK.target (LOCATABLE), PARTICIPATION.party (PARTY), PARTY_RELATIONSHIP.target (PARTY)
- CIMI RM version 1.0.12b (semver 1.0.0-dstu.1)
- Current solution:
  - all REF classes removed and **use associations**

What? Where? How?

What are LINKs?

- Reusable parts (like #include construct) – a reference (if specializations are to be allowed as well)
- Named links between instances – openEHR links
- Slots – items choosen from a set defined by queries or exclusions
- Can be substituted in a later model constraint step?
- Can be substituted at runtime

Scope

- Out-of-scope: instance/runtime refs
  - #include based – pattern / archetype_id / code; links between models ←designtime / modeling level
  - Slot / AOM
- In-scope: logical
  - Link to **type of instance** – e.g. archetype_id, pattern, code, subset expression ←restrict what hapens at runtime
    - RM

Where to solve?

- **RM Level:**
  - **CIMI RM ←*our focus now, what is needed here to support LINK/REF?***
- Archetype Level:
  - AOM & UML (AML Archetype Profile)

How?

- Quote from *"The Unified Modelling Language Reference Manual - Rumbaugh, Jacobson and Booch]. "At the logical level, we decide that we don't need to make a decision as to how these references are defined. However, in our instance examples we will need to choose one of the options for representing these associations."*
- Proposal:
  - CIMI RM = logical model, keep the way it is
  - In the Archetype if defined in UML use the <link/reference> stereotype to apply a reference
  - Tagged value referenced_archetype_id, referenced_archetype_code, etc.
  - Do something similar in AOM?

OPTION 1

- We consider the CIMI Reference Model to be a **logical model**, and "Each instance of an **association** is a tuple of references to objects." [quote from "The Unified Modelling Language Reference Manual - Rumbaugh, Jacobson and Booch]. At the logical level, we decide that we don't need to make a decision as to how these references are defined. However, in our instance examples we will need to choose one of the options for representing these associations.

OPTION 2

- We use node identifiers to define the target of the associations (e.g. ISO-13606 uses rc_ids, which can be used to identify instances of each Locatable node). Please note that the archetype_node_id is not able to be used as it identifies nodes in an archetype, rather than nodes in an instance.

OPTION 3

- We use association. It would be quite simple to use an XPath in an example instance if XML is used to represent these instances. However, I believe that Tom suggested against this approach, as it did not allow the target Locatable to be moved to a different part of the record.

OPTION 4

- We use URIs (or 'EHR_URI') to define the target of the associations. This could potentially be defined in such a way that the target Locatable could be moved, and the URI continue to resolve to the correct Locatable. If this approach is adopted, then we need to define and agree on what this URI format would look like. This is the option that we discussed during the last meeting.

OPTION 5 (GF)

- Take a more general approach and create a 'LINKS package' that defines a generic interface.
- An interface that takes into account all aspects that I showed last Thursday.
- In that interface we can accommodate OPTION 4 when we consider the URI a query.
- A query where a defined response is expected or not.
- At the archetype level we can constrain this interface as much as we like and use them in patterns.