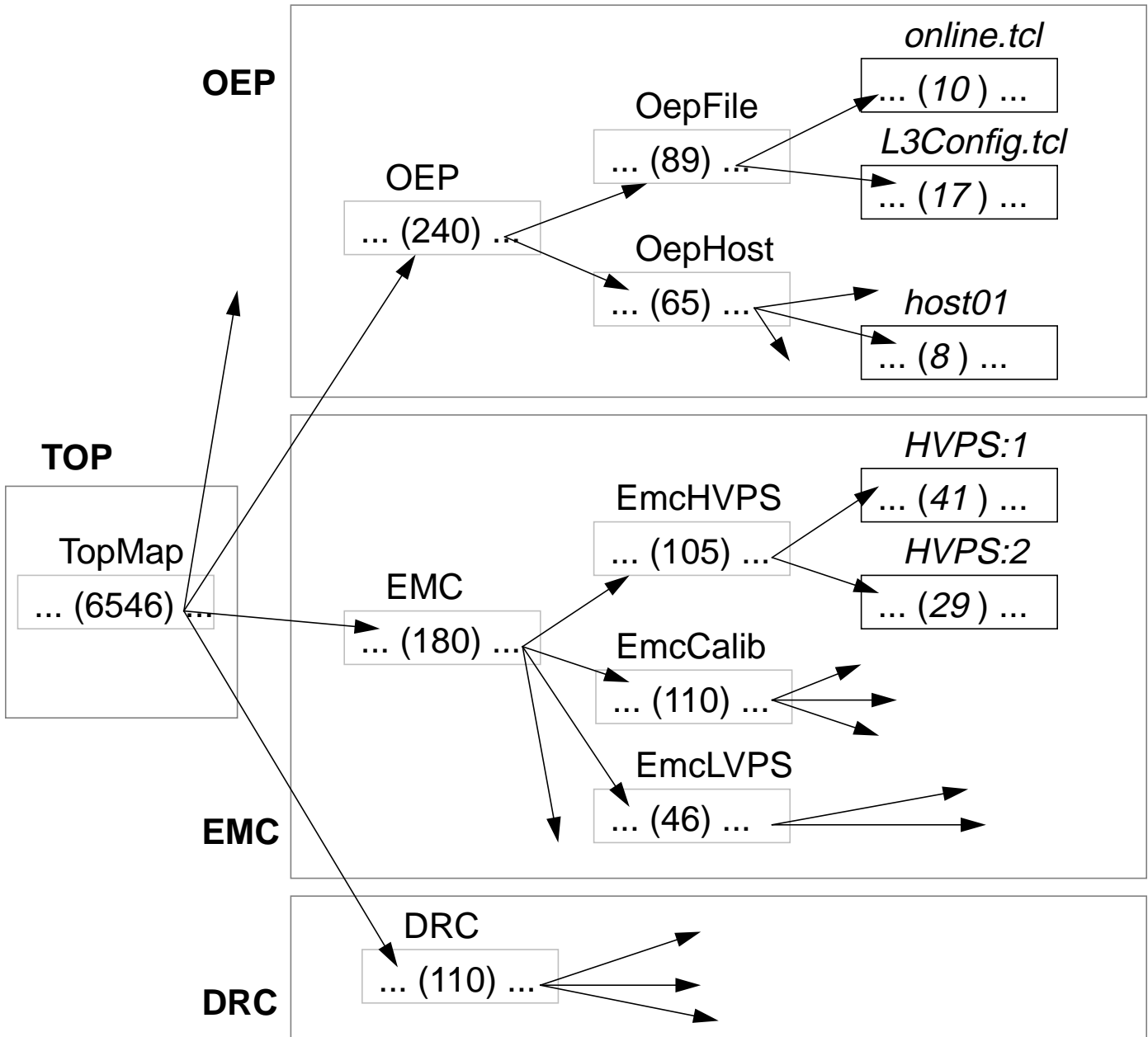


## Configuration database

- Part of the Online databases.
- It stores the configuration information of the various subsystems of the detector.
- Configuration objects are uniquely identified by the subsystem name, class name, secondary key and a 32-bit (d\_ULong) *Configuration Key*.
- All objects of a given subsystem are stored in a database named after the TLA of that subsystem.
- Objects of the same class are stored in a container named after the *class name* and a *secondary key*.
- All objects are assigned a *Configuration Key*, unique for the container. This key is incremented by 1 every time a new object is stored in the container.
- Configuration objects cannot be deleted or changed.
- Configuration objects can be stored directly in the Conditions database. Base class *BdbConfigObject* inherits from *BdbObject*.
- In the online environment the configuration data are primarily accessed by a *Global Configuration Key* generated by the Run Control. Configuration maps are used to build a *Configuration Tree* that translates the *Global Key* to the individual container-specific *Configuration Keys*. The branches of the tree are objects of the *BdbConfigMap* class while the leaves are the configuration objects. The *BdbConfigMap* class is a hash table that associates character labels (names) to persistent objects.

# Configuration Tree structure



□ Database

□ Map container

□ Object container

## Configuration database status

- Database implemented (Yury Kolomensky) and in use.
- Work for a server that interfaces the database to the VxWorks nodes is under way by Yury and David Brown.
- Need GUI database browsers and map editors to build the configuration tree.

## Ambient database

The Ambient database stores the readback values (ambient data) from the detector controls. Measurements are time-stamped and reported asynchronously to *Archiver* processes. The Archivers run continually. The stored data need to be available to users for online monitoring and offline analysis.

- The Conditions database was chosen (for now) as the ambient database.
- Since the data arrive at each Archiver process asynchronously they are accumulated in a transient conditions object over a period of half hour or until an END transition from the Run Control. The object is then stored in the Conditions database.
- The Conditions database interface and ProxyDict are used to stored and retrieve the data.
- A transient (persistent?) cache is needed to store the data while is being accumulated in the Archiver until it is stored in the database.
- The ProxyDict can hide the access of the data from the online users, retrieving them from their current location: either the Archiver's cache or the Conditions database.
- To access a given readback value based on the time T of its measurement, first the conditions object corresponding to that time T is retrieved from the database and then the object is searched to find the time-stamped value with time closer to T.

## Status of the Ambient database

- The Archiver process has been implemented using the Conditions database interface from release 6.4.2. It is available to subsystems for testing.
- The transient cache has not been implemented yet. The data is accessed directly through the Conditions database. If the nominal 30 minute interval is too long the Archivers can be configured to store more frequently.
- Need to modify the Archiver to use the new user interface to the Conditions database.
- Need to design and implement the Archiver cache and make it available to ProxyDict.