

SVN-repository
“nceplibs”
structural design
(draft 8)

Goal & Objectives

The goal:

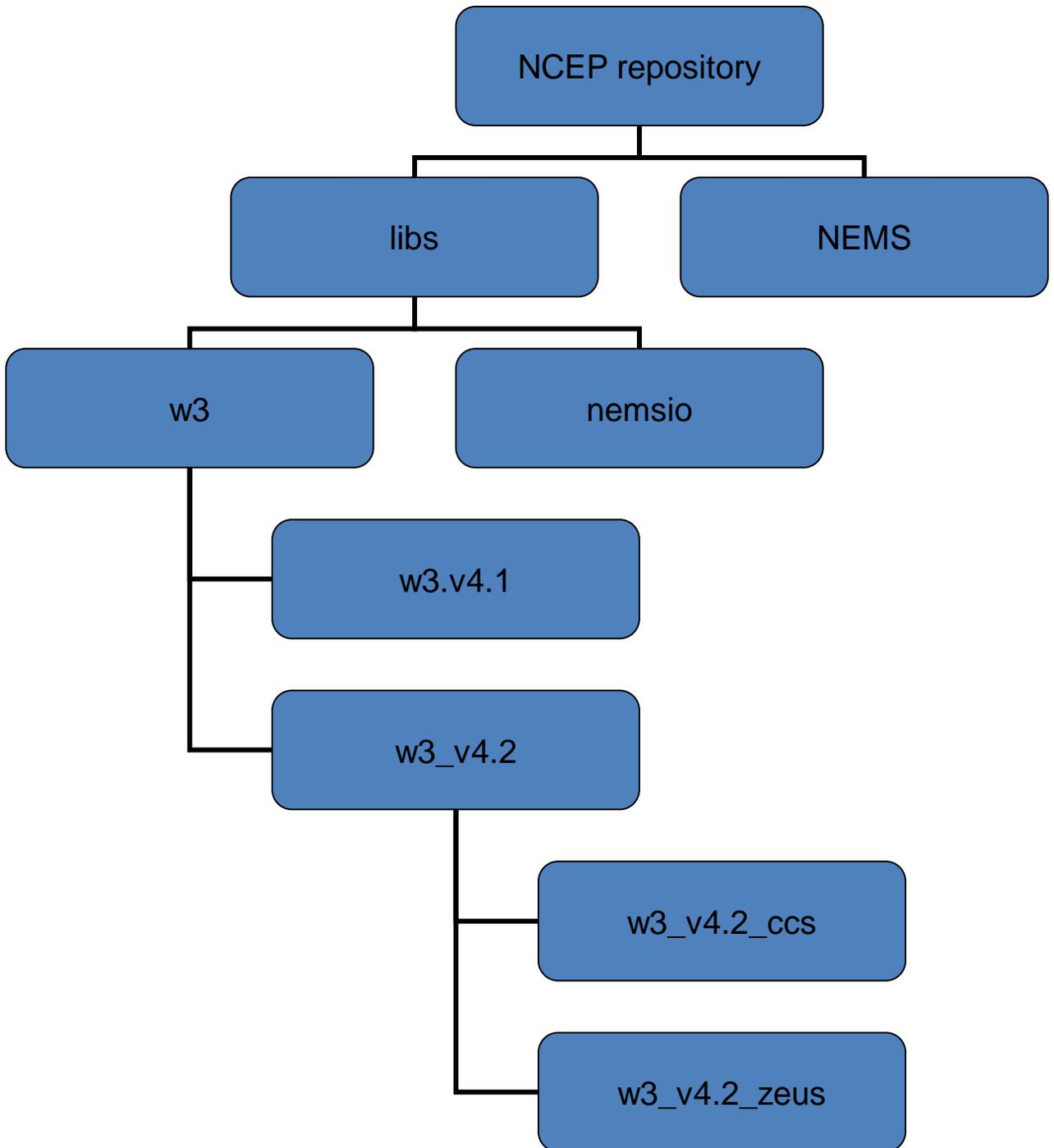
To provide NCEP developed libraries version control, updates and support procedures, serving related NCEP applications with a respect of the current libraries status and legacy of development.

Objectives:

To suggest a structure , answering on following questions:

1. Different people have different “versions” of the libs - where to put?
2. Different applications using different “versions” -- where to put?
3. Different “institutions” have diff versions -- where to put?
4. Different architectures and sites – where to put before integration?
5. “Equivalence” of the nceplib builds. Unit testing.
6. Releases – single tagging – where?
7. Complex (super-) tags – Application/site related – how to do?
8. Data pointers and association? Binary content

GENERAL INTENTION : To Organize



Version Control. Discussion.

What is a meaning of this structure

1. We realized, it's a superstructure: the entity is - a `_collection_` of nceplibs as well as a project name "nceplibs", So, checking out of `<svnemc>/nceplibs/trunk` shall give us all of the available libs - which can be quite rare use.
2. Also, a collection s of the libs are never developed all as a single project, but each single separately, so we decided to treat each single lib as a SVN sub-projects' concept, assuming that from a some point, a clean future development of each (after merging and fixing config (auto)) will be done (on for instance):

nceplibs/w3/trunk

nceplibs/w3/tags

nceplibs/w3/branches

and ci/co as a standard sub- project

3. In this case, the update of all libs-set doesn't need super-tags, but will be tagged on Nceplibs/Releases let say for NCO or DTC distributions, which will have a meaning of the certain working combination of the subprojects' status -- super-tags. The TAGS on the Releases will be planned and agreed with the operations upfront. That also may contain ready-to-use Apps-Arch related super-tags.
4. nceplibs/Branches/Apps + Platforms - really a place to collect and provide version control for NOW of all existent developments from different locations of the "versions". In most of the cases they're currently related with applications (GFS; GSI; NEMS..) ported on certain platforms - so it will be easier to collect this way. Needless to say that in the future, they are going to be dead and each sub-libs of the branch will be merge to the single lib development trunks... but it can take unpredictable time, while people do changes without version control. So, hopefully, it could be a temporal, but effective solution.

```

nceplibs/
  w3/
    branches/ /* development branches */
    tags/ /* singl lib tags/
    trunk/ /* Latest-current integrated singllel ib tags*/
  sigio/
    branches/
    tags/
    trunk/
  sp/
    branches/
    tags/
    trunk/
  nemsio/
    branches/
    tags/
    trunk/ ...
  branches/
    NCEP/
      Apps1_libs /*Application-related & site related
        ongoing development branch */
      Apps2_libs
        w3/
        sigio
        sp/
      DTC/ ...
      Apps1_libs
      Apps4_libs ...
    PORTS/
      Zeus
      Gaea
      Jet /* Ported, site-relevant common use libs
        placeholders)
      ...
  releases/ →(NCEP libs combinations TAGS)
    NCEP_libs_release1
    NCEP_libs_release2
    NCEP_libs_release3 /* Super-tagged supported releases (
    Apps_related_tags platform independent) and some
    milestones, including NCO for /nwprod */

```

How to make it workable?

1. Collect all available “private versions” of the libraries on the branches – to make it accessible in NCEP repository.
2. Research differences and eventually merge / integrate differences and built mechanisms on the single libs development trunks. .. nceplibs/<lib>/trunk
3. Establish a team with related responsibilities and support procedures.
4. Open EMC - nceplibs forum , allowing to discuss issues outside of the mailboxes.
5. Include into emc-helpdesk duties to collect and follow to the nceplibs’s change functionalities requests.
6. Provide periodic info on the patches and updates.
7. Agree on the plan of the releases and change management of the libs releases.
8. Stop develop “personalized libraries” or call “libraries” personal code development