Testing Java HLAs

These are the steps to verify core functionality of Java high level applications (HLA) when releasing new versions.

Version Control Migration

Any applications currently using CVS as their version control system should be migrated to Git as soon as possible using the `eco cvs2git` tool.

1. Run: (assumes proper AFS ACL permissions have been granted)

   ```
   $ eco cvs2git
   ```

2. Enter the name of the module as it appears in the `/afs/slac/g/lcls/cvs/CVSROOT/modules` file
3. Select the software subcategory (timing, mps, etc) and Git repo name for the migration
4. A bare Git repo will be generated under `$GITROOT` in the path you specified in step (3). (The previous CVS history and metadata will be migrated to the new Git repo automagically)
5. Verify that the new Git repo contains all of the previous CVS history and metadata by using the `eco` tool:

   ```
   $ cd /path/to/working/directory
   $ eco <module_name>
   $ cd /path/to/git/clone/repo
   $ git log
   ```

Building Java HLAs

Java HLAs are now deployed as static builds with their Java library dependencies built and checked in with the project (typically in a `lib` folder in the project's root directory).

Re-build order with new Java version:

1. Update JCA (EPICS extensions)/3rd party source
2. xal4lcls (include new jca.jar from previous step, update ojdbc5.jar from $ORACLE_HOME if needed)
3. MessageLogAPI (dependency for hlaCommon)
4. hlaCommon (update compile source and target in build.xml)
5. hlaExtensions (update compile source and target in build.xml)
6. APIs (MPSConfig, TimingPatternChooser, etc.)
7. HLAs

Related articles

- Testing Java HLAs