Building Community through xSDK Software Policies
(the slides are available under “Presentation Materials” in the above URL)
Date: December 11, 2019
Presented by: Ulrike Meier Yang (Lawrence Livermore National Laboratory) and Piotr Luszczek (The University of Tennessee, Knoxville)

Q. On slide 27 / Mandatory policy #4, what’s the procedure to get access to DOE machines for testing candidate packages on the DOE environments? For example do we request through XSEDE?

A. You can get access to systems at NERSC, OLCF, and the ALCF for this purpose through each facility’s Director Discretionary/Getting Started program.

- OLCF Directors Discretion Project Application
- ALCF Director’s Discretionary Allocation Program
- First NERSC Allocation

Q. M8 can version information be provided via a macro, or need it be an actual function call?

A. Yes it has to be a function call. This forces embedding of the version information in the library and allows validation of the binary files. A macro will only reveal information about the headers leaving the binaries in unknown state. Note: this policy does not apply to header-only libraries in C++.

Q. M9 how do you ensure the namespace prefixes are unique across the ecosystem? Simply prepending an arbitrary string with _ does not create global uniqueness. Have you considered a central registry of prefixes?

A. This may be an issue for very large collections of prefix names that require conflict resolution between packages with the same name. Java’s package naming is an example of trying to solve this problem at the Internet scale. We don’t quite have the same scale and the simple prefix worked so far. The only issue we encountered was too short name for the prefix: three characters long. We simply asked for the name to be extended which avoided name clashes.

A. Generally they are the library names, which are usually unique. In cases where the prefix was too simple, we have asked the developers to find a more unique one.
We have not considered a central registry (yet), but maybe we should. It is a good idea.

Q. For M13 is there a CMAKE equivalent of that policy?

A. cmake -DCMAKE_INSTALL_PREFIX=/path/to/install/ /path/to/package/source

Q. From an ECP ST perspective, should we be working with our 2.3.n SDK to ensure we are meeting the M and R policies? Or do we work with xSDK directly?

A. If your product is a math library, you should work with the xSDK, but if your product belongs to another category, I would suggest to work with your SDK.

Q. The current policy document appears to assume the use of MPI, and be written focused on the requirements of MPI-based packages. Have you considered policies for alternative NON-MPI programming models? ECP is funding several such alternative models.

A. So far we had no reason to look at other programming models, since all xSDK libraries and those considering xSDK membership are/were using MPI. It is something that would be interesting to discuss, especially if there are math libraries using a different programming model than MPI that are interested to join. Another relevant note is the node interoperability effort that we continue. We engaged representatives of various libraries and projects in ECP and the external ones. This is enable xSDK to accommodate these efforts at the node level: think projects with the word “open” in the name, the vendor software stacks, runtime systems, and communication libraries. As soon as we reach a consensus we could start drafting policies that enable on-node integration.