# What are Interoperable Software Libraries?

## Introducing the xSDK

The IDEAS Scientific Software Productivity Project
ideas-productivity.org/resources/howtos/

**Background:** As extreme-scale computational science increasingly incorporates multiscale and multiphysics modeling, simulation, and analysis, the combined use of software developed by independent groups has become imperative: no single team has resources for the full range of capabilities needed for predictive science and decision support. **Software libraries** have proven effective in providing widely reusable software that is robust, efficient, and scalable for high-performance computing (HPC). Moreover, scientific application codes can employ library design principles to help manage complexity and achieve good performance, whether the application software is intended for use in a single context or modest reuse across applications in the same domain (e.g., as **domain components**). While the following discussion uses terminology of software library interoperability, the concepts also apply to application-specific domain components.

**A software library** is a high-quality, encapsulated, documented, tested, and multiuse software collection that provides functionality commonly needed by application developers. Key advantages of software libraries include leverage of library developer expertise and reduced application coding effort. For example, numerical software libraries provide easy access to sophisticated mathematical algorithms and high-performance data structures that have been developed by experts, so that application users do not need to write this complex code and can instead focus on their scientific domain software.

Libraries can provide control inversion via abstract interfaces, call-backs, or similar techniques such that user-defined functionality can be invoked by the library, for example, a user-defined sparse matrix multiplication routine. Libraries can also facilitate construction of related specific objects that provide customizable behavior to improve performance or flexibility. Moreover, libraries can include domain-specific software components that are designed to be used by more than one application.

**Software library interoperability** refers to the ability of two or more libraries to be used together in an application code, without special effort on the part of the user. For simplicity, we discuss interoperability between two libraries; extension to interoperability among three or more libraries is conceptually straightforward. Depending on application needs, various levels of interoperability can be considered:

- **Interoperability level 1:** both libraries can be used (side by side) in an application
- **Interoperability level 2**: the libraries can exchange data (or control data) with each other
- **Interoperability level 3**: each library can call the other library to perform unique computations

This material is based upon work supported by the U.S. Department of Energy Office of Science, Advanced Scientific Computing Research and Biological and Environmental Research programs.

The simplest case (interoperability level 1) occurs when an application needs to call two distinct libraries for different functionalities (for example, an MPI library for message-passing communication and HDF5 for data output). As discussed in [1, 2], even this basic level of interoperability requires consistency among libraries to be used within the same application, in terms of compiler, compiler version/options, and other third-party capabilities. If both libraries have a dependence on a common third party, the libraries must be able to use a single common instance of it. For example, more than one version of the popular SuperLU linear solver library exists, and interfaces have evolved. If two libraries both use SuperLU, they must be able to work with the same version of SuperLU. In practice, installing multiple independently developed packages together can be a tedious trial-and-error process.

Interoperability level 2 builds on level 1 by enabling conversion, or encapsulation, and exchange of data between libraries. This level can simplify use of libraries in sequence by an application. In this case, the libraries themselves are typically used without internal modification to support the interoperability.

Interoperability level 3 builds on level 2 by supporting the use of one library to provide functionality on behalf of another library. This level of interoperability provides significant value to application developers because they can access capabilities of additional libraries through the familiar interfaces of the first library.

## The Extreme-Scale Scientific Software Development Kit (xSDK)

A key aspect of work in the IDEAS project is development of the Extreme-scale Scientific Software Development Kit (xSDK)—a collection of related and complementary software elements that provide the building blocks, tools, models, processes, and related artifacts for rapid and efficient development of high-quality applications.

**xSDK community policies:** The xSDK addresses interoperability among the high-performance numerical libraries hypre, PETSc, SuperLU, and Trilinos.  The xSDK ensures level 1 interoperability for each xSDK library via a full-featured build script and testing environment and a collection of community policies. The following draft xSDK community policies address challenges in interoperability level 1.

- **xSDK package community policies:** ***A set of required policies*** (including topics of configuring, installing, testing, use of MPI, portability, contact and version information, open source licensing, namespacing, and repository access) that a software package must satisfy in order to be considered **xSDK compatible**. This designation informs potential users that the package can be easily used with other xSDK libraries and components. Also presented are ***recommended policies*** (including topics of public repository access, error handling, freeing system resources, and library dependencies), which are encouraged but not required. Similarly, a package can become an **xSDK member package** if (1) it is an xSDK-compatible package, *and* (2) it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

This material is based upon work supported by the U.S. Department of Energy Office of Science, Advanced Scientific Computing Research and Biological and Environmental Research programs.

- **xSDK community installation policies: GNU Autoconf and CMake options**: ***A standard subset of configure and CMake options for xSDK and other HPC packages*** in order to make the configuration and installation as efficient as possible on standard Linux distributions and Mac OS, as well as on target machines at DOE computing facilities (ALCF, NERSC, and OLCF).

The xSDK collection of software packages commits to adhere to these community policies in order to ensure compatibility with other packages that meet the same standards. The aim is to simplify the combined use of multiple independently developed software packages and to provide a foundation for addressing broader issues in interoperability and performance portability.

**Deeper levels of xSDK interoperability** involve exchanging, controlling, and interpreting data, as well as calling routines between libraries (interoperability levels 2 and 3 described above). Initial xSDK capabilities of hypre, PETSc, SuperLU, and Trilinos support interoperability among scalable linear solvers, so that applications can readily experiment with algorithms across multiple packages, in combination. Forthcoming companion documents will explain approaches used for interfaces and adapters between packages as well as work on interoperability layers for other functionalities. A longer-term goal is collaboration among members of the HPC community to improve software interoperability as needed by extreme-scale computational science.

**References:**

**[1]** Package Management Practices Essential for Interoperability: Lessons Learned and Strategies Developed for FASTMath, M. C. Miller, L. Diachin, S. Balay, L. C. McInnes, and B. Smith, First Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE), Nov 17, SC13.

**[2]** Smart Libraries: Best SQE Practices for Libraries with Emphasis on Scientific Computing, M. C. Miller, J.F. Reus, R.P. Matzke, Q.A. Koziol, A.P. Cheng, Proceedings of the Nuclear Explosives Code Developer's Conference, Dec 2004.

This document was prepared by Lois Curfman McInnes, Michael Heroux, Xiaoye Li, Barry Smith, and Ulrike Yang, with contributions from all xSDK developers.

This material is based upon work supported by the U.S. Department of Energy Office of Science, Advanced Scientific Computing Research and Biological and Environmental Research programs.