



Part I: Documentation



The coupling library for partitioned multi-physics simulations

Documentation version 3.3.1

Generated: March 13, 2026

preCICE is free/open-source software, using the GNU LGPL3 license. The code is publicly available and actively developed on Github at <https://github.com/precice/precice>.

This pdf document is a snapshot of the preCICE documentation hosted at precice.org as of March 13, 2026. The HTML/CSS version of the documentation is available on Github at <https://github.com/precice/precice.github.io> and can also be built locally with Jekyll. For more information consult the [README](#) in this repository.

Table of Contents

Documentation

Fundamentals

Overview.....	6
Terminology.....	8
License information.....	10
Literature guide.....	12
Roadmap.....	16
FAQ.....	18
Previous versions.....	19

Installation

Overview.....	20
System packages.....	23
Using Spack.....	25
Building from source	
Preparation.....	28
Dependencies.....	30
Configuration.....	39
Building.....	42
Testing.....	43
Installation.....	44
Finding.....	45
Advanced.....	47
Troubleshooting.....	49
Notes on CMake.....	52
Linking to preCICE.....	53
Language bindings	
Fortran.....	55
Python.....	56
Matlab.....	57
Julia.....	58
Rust.....	60
Special systems.....	61
Demo Virtual Machine.....	75
preCICE distribution.....	78

Configuration

Overview.....	81
Basics	
Introduction.....	82
Mapping.....	85

Communication.....	92
Coupling scheme	94
Acceleration.....	97
Mesh exchange example	101
Advanced topics	
Multi coupling	103
Logging.....	105
Export.....	108
Action	112
Watchpoint.....	115
Watch integral	116
XML reference.....	117
Tooling	
Overview.....	194
Artificial solver testing environment (ASTE)	195
The preCICE CLI	202
Built-in tooling.....	204
Config visualization.....	206
FMI runner	211
Micro Manager	
Overview.....	215
Get the Micro Manager.....	217
Preparing micro simulation	219
Configuration	222
Running.....	227
Logging.....	228
Adaptivity	229
Model adaptivity.....	232
Snapshot Computation	235
Performance analysis.....	238
RBF shape calculator.....	245
Provided adapters	
Overview.....	246
CalculiX	
Overview.....	249
Get CalculiX.....	250
Get the adapter.....	253
Build the adapter with PaStiX	255
Configuration	259
Troubleshooting	265
Building on SuperMUC	266
code_aster	
Overview.....	268
deal.II	

Overview	274
Get the adapter	275
Configuration	278
For your own deal.II code	282
Limitations and assumptions	284
Solver details	285
Coupling Meshes in deal.II	287
DuMuX	
Overview	288
Get the adapter	289
Use the adapter	291
DUNE	
Overview	292
FEniCS	
Overview	293
G+Smo	
Overview	295
Nutils	
Overview	297
OpenFOAM	
Overview	298
Get the adapter	301
Configuration	304
Extending	317
OpenFOAM support	320
SU2	
Overview	328
Get the adapter	329
Configuration	331
Couple your code	
Overview	334
Application programming interface	335
Step by step	
Step 1 – Preparation	337
Step 2 – Steering methods	338
Step 3 – Mesh and data access	340
Step 4 – Coupling flow	344
Step 5 – Non-matching time step sizes	347
Step 6 – Implicit coupling	353
Step 7 – Data initialization	356
Step 8 – Mesh connectivity	357
Step 9 – Gradient Data	362
Advanced topics	
Adapter software engineering	365

Initialization in existing MPI environment	369
Moving or changing meshes	370
Just-in-time data mapping.....	372
Dealing with FEM meshes.....	376
Dealing with distributed meshes.....	378
Direct access to received meshes	382
Time interpolation of coupling data.....	384
Global data	390
Porting guides for major versions	
Porting adapters in general	391
Porting from 1.x to 2.x.....	392
Porting from 2.x to 3.x	394
Running simulations	
Overview.....	401
Running locally	402
Distributed systems.....	405
SLURM sessions	407
Output files	410
Dev docs	
Overview.....	414
Naming.....	415
General coding conventions.....	417
Timings in preCICE.....	419
Logging.....	421
Optimization.....	424
Release workflow	425
Running and writing tests	426
System tests	430
Tooling	437
Release strategy	439
Publication strategy.....	441
Tutorials	
<hr/>	
Introduction	
Overview.....	446
Visualization	449
Basic cases	
Quickstart.....	450
Flow over heated plate	457
Partitioned heat conduction	462
Perpendicular flap.....	466
All tutorials	
ASTE turbine	475

Breaking dam with flexible pillar 2D.....	478
Channel transport	
Basic variant.....	480
Reaction.....	483
Particles.....	485
Elastic tube 1D.....	487
Elastic tube 3D.....	492
Flow around controlled moving cylinder.....	495
Flow over a heated plate	
Basic variant.....	457
Nearest-projection mapping.....	504
Steady state.....	507
Two meshes.....	510
Free flow over porous media.....	512
Heat exchanger.....	517
Heat exchanger simplified.....	521
Multiple perpendicular flaps.....	524
Oscillator.....	528
Oscillator overlap.....	531
Partitioned elastic beam.....	533
Partitioned flow	
Partitioned flow over a b.f. step.....	535
Partitioned flow over heated plate.....	539
Partitioned pipe.....	542
Partitioned pipe two-phase.....	546
Partitioned heat conduction	
Basic variant.....	462
Complex variant.....	553
Direct mesh access.....	555
Overlapping Schwarz.....	557
Perpendicular flap	
Basic variant.....	466
Stress variant.....	568
Quickstart.....	450
Resonant circuit.....	578
Two-scale heat conduction.....	581
Turek-Hron FSI3.....	585
Volume-coupled diffusion.....	588
Volume-coupled flow.....	590

The preCICE documentation

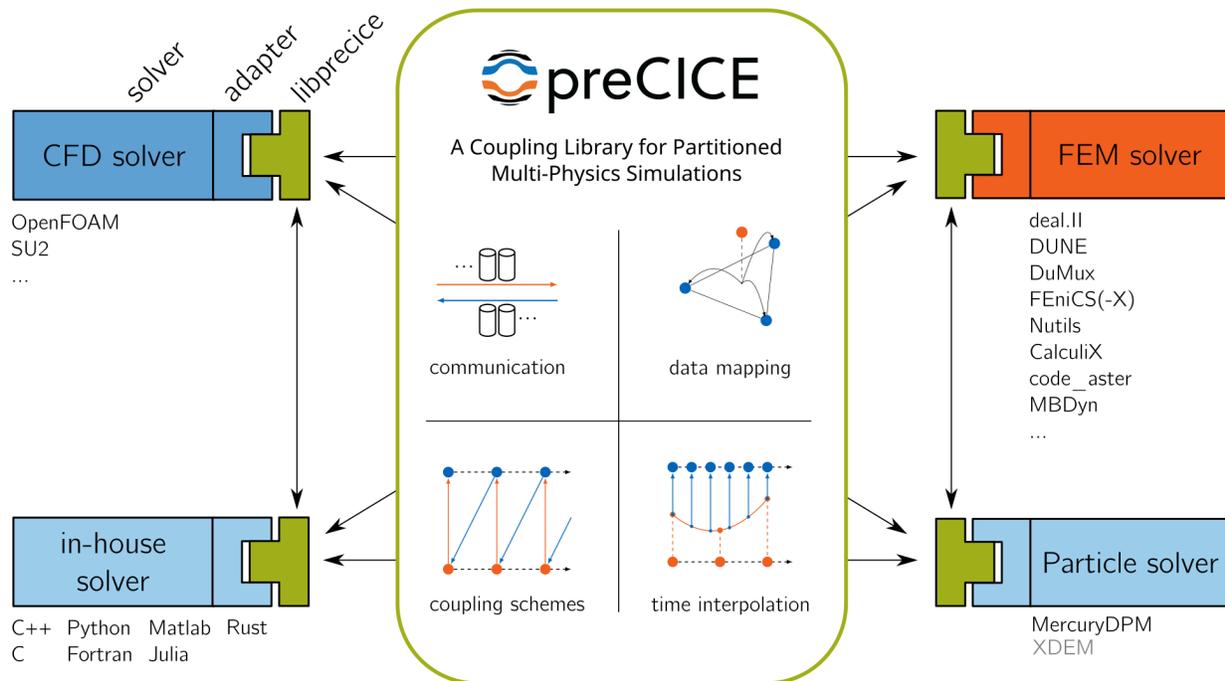
Summary: This page gives an overview of the complete preCICE documentation, including building, configuration, literature, the API, and much more.

The big picture

preCICE stands for Precise Code Interaction Coupling Environment. Its main component is a library that can be used for partitioned multi-physics simulations, including, but not restricted to fluid-structure interaction and conjugate heat transfer simulations. Partitioned (as opposite to monolithic) means that preCICE couples existing programs (solvers) which simulate a subpart of the complete physics involved in a simulation. This allows for the high flexibility that is needed to keep a decent time-to-solution for complex multi-physics scenarios, reusing existing components. preCICE runs efficiently on a wide spectrum of systems, from low-end laptops up to complete compute clusters and has [proven scalability \(page 14\)](#) on 10000s of MPI Ranks.

The preCICE library offers parallel communication means, data mapping schemes, and methods for transient equation coupling. Additionally, we are actively developing methods for time interpolation and more features (see our [roadmap \(page 16\)](#)). preCICE is written in C++ and offers [additional bindings \(page 335\)](#) for C, Fortran, Python, Rust, Julia and Matlab. Coupling your own solver is very easy, due to the minimally-invasive approach of preCICE. Once you add the (very few) calls to the preCICE library in your code, you can couple it with any other code at runtime. For well-known solvers such as OpenFOAM, deal.II, FEniCS, Nutils, CalculiX, or SU2, you can use one of our official adapters.

preCICE is free/open-source software, using the [GNU LGPL3 license](#). This license ensures the open future of the project, while allowing you to use the library also in closed-source solvers. The code is publicly available and actively developed on [GitHub](#). See more [license information \(page 10\)](#).



Writing about preCICE? [Get this image and more material](#).

Where to find what

This documentation explains how to use preCICE. We do not detail the numerical methods and HPC algorithms in the preCICE docs, but we refer to existing publications on preCICE for these topics. The [literature guide \(page 12\)](#) gives an overview of the most important preCICE literature.

The preCICE docs are organized in several sections:

- [Installation \(page 20\)](#): How to get and install preCICE on various systems.
- [Configuration \(page 81\)](#): At runtime, preCICE needs to be configured with an xml file. Here you learn how to do that.
- [Tooling \(page 194\)](#): Several helpful (but completely optional) tools around preCICE: tools for setting up your simulation, post-processing the results, and much more.
- [Provided adapters \(page 246\)](#): The preCICE community maintains ready-to-use adapters for many popular solvers. Here, you find the documentation of these adapters.
- [Running simulations \(page 401\)](#): Learning how to run preCICE simulations on various types of machines.
- [Couple your code \(page 334\)](#): Getting familiar with the preCICE API.
- [Dev docs \(page 414\)](#): References that developers use. Are you maybe also thinking of [contributing \(page 0\)](#)?

Before you start reading: there are just some [preCICE-specific technical terms \(page 8\)](#) that every user should read first.

☑ **Tip:** Interested in training? We have developed a [training course \(page 0\)](#) on preCICE. Come to the yearly [preCICE workshops \(page 0\)](#) or book a private training through the [support program \(page 0\)](#).

Terminology

Summary: We often refer to the following terms, but they may not already be clear.

Partitioned approach

As already mentioned in the overview:

Partitioned (as opposed to monolithic) means that preCICE couples existing programs (solvers) which simulate a subpart of the complete physics involved in a simulation.

The direct opposite is the (numerically) monolithic approach, in which the same software has to construct and solve a global system of equations for the complete domain.

There are several advantages and disadvantages in both approaches. The partitioned approach allows to reuse existing components, reducing the time from deciding to simulate a multi-physics scenario to getting accurate results (the real time to solution). It also allows to study different combinations of components that are already “experts” in each subdomain. The monolithic approach can have robustness and performance advantages in some cases, but with the current advanced partitioned coupling algorithms, the significance of any such difference should not always be taken for granted (see our [literature guide \(page 12\)](#)).

Solver and participant

By *solver*, we refer to a complete simulation code, which we want to couple. We do not mean a linear algebra solver. With *participant*, we refer to a solver in the context of a coupled simulation (e.g. “Fluid participant”). This term is also used in the [preCICE configuration \(page 81\)](#).

Library approach

preCICE follows a *library* approach. This basically means that preCICE is a library: each solver needs to call preCICE. This also means that preCICE runs in the same threads that the solvers run in. The opposite coupling approach is the *framework* approach. In that approach, the coupling tool calls all solvers, which need to implement a certain programming interface. The advantage of a library approach is that it is minimally invasive to the coupled codes. They do not need to be rewritten. Instead, you just need to insert the preCICE calls at the right places.

Peer-to-peer approach

preCICE also follows a peer-to-peer approach. If you already tried preCICE, you may have noticed that you only need to start all coupled solvers individually, in the same way you would start them to run each single-physics simulation. There is no other starting mechanism involved: no server-like coupling executable or anything similar.

Adapter

To call preCICE from your code, you need to call functions of the application programming interface of preCICE. You can directly do this in your code. In this case, you develop an adapted solver. The little software engineering purist in you prefers, however, to collect all calls to preCICE into one place. This could be a separate class or module in your code. This could also be a separate library, which you call from pre-defined callback hooks. We call this one place an *adapter*. Depending on the perspective, you would call it *preCICE adapter*, *MyCode adapter*, or *MyCode-preCICE adapter*; assuming that you want to couple a code named MyCode. [preCICE comes with a few ready-to-use adapters \(page 246\)](#). If you want to couple your own code, you basically want to develop an adapter for this code. [Read more on adapter software engineering approaches \(page 0\)](#).