System Coupling User's Guide

Ansys
ANSYS, Inc.
Southpointe
2600 Ansys Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Release 2025 R1
January 2025

Copyright and Trademark Information

Disclaimer Notice

U.S. Government Rights

Third-Party Software

Published in the U.S.A.

Table of Contents

List of Figures

List of Tables

Introduction to the *System Coupling User's Guide*

The *System Coupling User's Guide* describes how to use the current version of ***Ansys System Coupling*** to direct two or more otherwise independent physics solvers and/or external data sources to work together in a coupled multiphysics analysis — for example, in a Fluid-Structure Interaction (FSI) analysis.

System Coupling User's Guide Contents

This guide covers the following topics:

- System Coupling Overview (p. 19) offers a high-level overview of the core coupling functionality available for all analyses run using System Coupling, including troubleshooting information, migration instructions, and a listing of current known issues and limitations.
- Using System Coupling's User Interfaces (p. 53) describes how to set up and execute coupled analyses using System Coupling's graphical user interface (GUI) and command-line interface (CLI).
- Using System Coupling in Workbench (p. 237) describes how to set up and execute coupled analyses using System Coupling in Workbench.
- Coupled Analysis Management (p. 277) describes the processes System Coupling uses to manage coupled analyses and co-simulations.
- System Coupling Data Transfers (p. 281) provides detailed and context-agnostic information about data transfers, including the rules that govern their creation, the overall data-transfer process, and details about the mapping and supplemental processing algorithms used by System Coupling.
- System Coupling Input and Output Files (p. 311) describes the files that System Coupling consumes and produces during the setup and execution of a coupled analysis.
- Common Engineering Applications for System Coupling (p. 371) provides information on some of the different types of analyses supported by System Coupling.
- Best Practices for System Coupling (p. 393) provides both generally applicable and application-specific best practices and recommendations for setting up and running coupled analyses.

Additional Resources

In addition to this guide, the following System Coupling documentation is available:

- *System Coupling Settings and Commands Reference* provides a comprehensive listing of System Coupling's data model settings, commands, built-in variables, and command-line options.
- *System Coupling Beta Features* gives instructions for accessing and using the hidden features offered by System Coupling in the 2025 R1 release.

- *System Coupling Tutorials* includes tutorials that provide guidance on using System Coupling, demonstrating its use in specific co-simulation applications.
- *System Coupling Participant Library* documentation for System Coupling's C, C++, Fortran, and Python APIs is available on the Ansys Developer Portal site. It includes reference materials, examples, tutorials, and instructions on how to use APIs to connect independent solvers to a System Coupling co-simulation.
- PySystemCoupling Documentation provides guidance on using PySystemCoupling, an interface that lets you use System Coupling within or alongside any Python environment, whether in conjunction with other Ansys Python libraries and packages or with other external Python products.

System Coupling Overview

The Ansys portfolio of simulation software facilitates the creation of multidisciplinary physics analyses — not only within the context of a single product, but also through the use of ***Ansys System Coupling***. System Coupling can integrate multiple individual analyses, enabling you to leverage different physics solvers and/or static external data sources in a single multiphpysics simulation. When two or more analyses are coupled, an examination of their combined results can capture more complex interactions than an examination of those results in isolation, producing more accurate results and yielding an optimal solution.

You can use System Coupling in several different ***contexts***, each of which facilitates comprehensive multidisciplinary simulations between coupling participants. You can run System Coupling in:

- the System Coupling's graphical user interface (**GUI**)
- the System Coupling's command-line interface (**CLI**)
- System Coupling in Workbench (**WB**)

System Coupling manages the execution of simulations between *coupling participants*, which are the applications or data sources that send and/or receive data in a coupled analysis. Participant support is determined by the context you're using.

This chapter provides a high-level introduction to core System Coupling functionality available in all three contexts. For more information, see:

The Ansys Product Improvement Program

Ansys System Coupling is covered by the Ansys Product Improvement Program (APIP), which enables Ansys, Inc. to collect and analyze anonymous usage data reported by our software without affecting your work or product performance. Analyzing product usage data helps us to understand customer

usage trends and patterns, interests, and quality or performance issues. The data enable us to develop or enhance product features that better address your needs.

For details on the program as a whole, see The Ansys Product Improvement Program in the *Mechanical User's Guide* (or the corresponding section in the documentation for another Ansys product covered by the program).

APIP Participation Preferences

Ansys System Coupling respects the participation preferences you've set in other Ansys products. If, when starting System Coupling, you have not yet specified a preference in another product, System Coupling assumes that you have not agreed to participate and does not collect data. If you wish to change your existing participation preference, access the **Ansys Product Improvement Program** dialog from a participant product's **Help** menu.

Data Collected by Ansys System Coupling

If you agree to participate in the APIP program, then System Coupling collects the coupling-specific data listed in the table below.

Table 1: Data collected by System Coupling for the Ansys Product Improvement Program

Category	Data collected	Description
APIP Data	Country code	Code for the country in which APIP data is collected
	Time zone	Time zone in which APIP data is collected
System Data	OS	Operating system used
	Graphics card	Graphics card used
	Cores	Number of cores available for the coupled analysis
Coupling Participant Data	Number of coupling participants	Total number of coupling participants included in the coupled analysis
	Participant types	Ansys products engaging as coupling participants
	Number of data transfer sources	Number of coupling participants sending data to another participant in a data transfer
	Source names	List of all Ansys products sending data to another participant
	Number of data transfer targets	Number of coupling participants receiving data from another participant in a data transfer
	Target names	List of all Ansys products receiving data from another participant
System Coupling Data	Application ID	Identifier for the System Coupling application
	Application version	Version of System Coupling used

Category	Data collected	Description
	Session ID	Unique identifier for the System Coupling session
	Coupling Platform	Whether System Coupling runs in one of its standalone user interfaces or in Workbench
	Coupling Mode	Whether System Coupling runs in co-simulation or mapping mode
	Coupling Context	Whether System Coupling runs in its GUI or CLI
Coupled Analysis Data	Analysis type	Whether System Coupling runs a Steady, Transient, or mixed Steady-Transient solution
	Stabilization	Type of solution stabilization applied, when applicable
	Number of data transfers	Total number of data transfers defined for the coupled analysis
	Number of mesh nodes	Total number of mesh nodes on all coupled regions
	Number of mesh elements	Total number of mesh elements on all coupled regions
	Maximum number of interface regions	Total number of regions on the interface side with the most regions
	Cores	Number of cores used for the coupled analysis (as specified by the `-s` command-line argument)
	Solve time	Duration of the solution's execution

Coupled Analysis Licensing Requirements

The licenses needed for System Coupling analyses are listed in Table 2: Applicable licenses for participants in System Coupling analyses (p. 21)

To run coupled analyses using System Coupling, you need either one license for each participant solver or a **Multiphysics Licensing Bundle** (shown in bold text in the table below). No additional license is needed for the System Coupling infrastructure itself.

System Coupling runs a license check the first time a command which starts the participants is called. If the license check fails, an error message is displayed. Ensure that at least one of the licenses listed in the table is available before starting the participants again.

Ansys EnSight is required for the postprocessing of System Coupling's interface results. If you are using a Multiphysics Licensing Bundle that does *not* include an EnSight license (shown in **italicized** text in the table below), you will need a separate EnSight license to visualize co-simulation results.

Table 2: Applicable licenses for participants in System Coupling analyses

Product	Applicable licenses
Ansys CFX	• **Ansys Mechanical CFD Maxwell**

Product	Applicable licenses
	• **Ansys Mechanical CFD** • Ansys CFD Enterprise • Ansys CFD Enterprise Solver • Ansys CFD Premium • Ansys CFD Premium Solver
Ansys Fluent	• **Ansys Mechanical CFD Maxwell** • **Ansys Mechanical CFD** • Ansys CFD Enterprise • Ansys CFD Enterprise Solver • Ansys CFD Premium • Ansys CFD Premium Solver
Ansys Forte	• **Ansys Mechanical CFD Maxwell** • **Ansys Mechanical CFD** • Ansys CFD Enterprise • Ansys CFD Enterprise Solver • Ansys CFD Premium • Ansys CFD Premium Solver • Ansys Chemkin Enterprise
Ansys Maxwell	• **Ansys Mechanical CFD Maxwell** • ***Ansys Mechanical Maxwell*** • Ansys Electronics Enterprise • Ansys Electronics Enterprise Solver • Ansys Electronics Premium Maxwell
Ansys Mechanical	• **Ansys Mechanical CFD Maxwell** • **Ansys Mechanical CFD** • ***Ansys Mechanical Maxwell*** • Ansys Mechanical Enterprise

Product	Applicable licenses
	• Ansys Mechanical Enterprise Solver

System Coupling Capabilities

This section provides information on System Coupling's capabilities and participant-specific support of available coupling functionality. For more information, see:

System Coupling Capabilities by Context

Participant Variables and Quantity Types Supported by System Coupling

Participant Support of System Coupling Capabilities

System Coupling Capabilities by Context

Table 3: System Coupling Capabilities by Context (p. 23) lists the key coupling capabilities available for System Coupling's graphical user interface (**GUI**), System Coupling's command-line interface (**CLI**), and System Coupling in Workbench (**WB**).

Capabilities are divided into the following categories:

- User Environment (p. 23)
- Coupling Participants (p. 24)
- Co-Simulation (p. 25)
- Interfaces and Data Transfers (p. 25)
- Coupled Run Execution (p. 26)
- Co-Simulation Output (p. 27)

Table 3: System Coupling Capabilities by Context

Category		Capabilities	GUI	CLI	WB
User Environment	**User Interface**	Graphical user interface	✓		✓
		Optimized, scriptable command-line interface	✓	✓	
		Interactive solve commands	✓	✓	
		Command-line arguments for: • Coupling execution • Debugging • Parallel processing	✓	✓	
		Built-in solution and region variables	✓	✓	

Category		Capabilities	GUI	CLI	WB
		Participant APIs for connecting to System Coupling: [1 (p. 28)] • C • C++ • Fortran • Python	✓	✓	
	Expressions	Vector and scalar expressions	✓	✓	
		Named expressions	✓	✓	
		External Python functions in expression definitions	✓	✓	
		Settings with immutable real or integer single values	✓	✓	
		Settings with mutable single values (currently only time step size)	✓	✓	
		Settings with mutable field values (currently only source-side data transfer variables)	✓	✓	
Coupling Participants	**Supported Participants**	Ansys External Data file			✓
		Electronics Desktop	✓	✓	
		CFX	✓	✓	✓
		Fluent	✓	✓	✓
		Forte	✓	✓	
		Mechanical	✓	✓	✓
		Mechanical Server[5 (p. 28)]	✓	✓	
		CFD Server[5 (p. 28)]	✓	✓	
		Co-simulation Functional Mock-up Unit (FMU) participants	✓	✓	
		SCDT Server	✓	✓	
	Participant Dimension	2D[6 (p. 28)]	✓	✓	
		3D	✓	✓	✓
	Participant Setup	Check for changes to participant input file[3 (p. 28)]	✓	✓	
		Update participant after input file modifications[3 (p. 28)]	✓	✓	
		Participant instancing[3 (p. 28)]	✓	✓	
	Execution Controls	Customizable solver input files	✓	✓	

Category		Capabilities	GUI	CLI	WB
		Automated definition of participant working directories relative to the co-simulation directory	✓	✓	
		Additional command-line arguments for participant executable	✓	✓	
	Region Topologies	Surfaces	✓	✓	✓
		Volumes	✓	✓	
	Variable Properties	Extensive and intensive variables	✓	✓	✓
		Scalar and vector tensor types	✓	✓	✓
		Real numbers	✓	✓	✓
		Complex numbers	✓	✓	
		Real and integer attributes	✓	✓	
		Dimensionality	✓	✓	
Co-Simulation	**Number of Participants**	Up to two participants in a single co-simulation			✓
		Up to 25 participants in a single co-simulation	✓	✓	
	Analysis Types	Steady coupled analyses	✓	✓	✓
		Transient coupled analyses	✓	✓	✓
		Mixed steady-transient coupled analyses	✓	✓	✓
	Analysis Setup	Step-based analyses with multiple iterations per coupling step (for implicit couplings)	✓	✓	✓
		Set minimum and maximum number of iterations per step	✓	✓	✓
		Iteration-based analyses[3 (p. 28)]	✓	✓	
		Data model optimization for one-way workflows	✓	✓	
		Export coupling setup from Workbench			✓
Interfaces & Data Transfers	**Interfaces**	Multiple regions per interface side[3 (p. 28)]	✓	✓	
		Automatic detection of same interface sides for thin bodies[3 (p. 28)]	✓	✓	
		Automatic rigid-body interface alignment for surface-to-surface mapping	✓	✓	
		Geometry transformations for models with different orientations per interface side	✓	✓	
		Cylindrical geometry instancing per interface side	✓	✓	
	Data Transfers	Suppression of data transfers	✓	✓	✓

Category		Capabilities	GUI	CLI	WB
		Creation of data transfer groups	✓	✓	
	Transfer Topologies	Surface ⇔ Surface transfers	✓	✓	✓
		Volume ⇔ Volume transfers	✓	✓	
		Volume ⇔ 2D Surface transfers	✓	✓	
		Single Scalar Value ⇔ Surface / Volume transfers (for FMU participants)	✓	✓	
		Single Scalar Value ⇔ Single Scalar Value transfers (for FMU participants)	✓	✓	
	Transfer Quantities [4 (p. 28)]	Convection Reference Temperature	✓	✓	✓
		Electrical Conductivity	✓	✓	
		Force	✓	✓	✓
		Heat Rate	✓	✓	✓
		Heat Transfer Coefficient	✓	✓	✓
		Incremental Displacement	✓	✓	✓
		Mode Shape	✓	✓	
		Temperature	✓	✓	✓
	Mapping	Profile-preserving mapping algorithms	✓	✓	✓
		Locally and globally conservative mapping algorithms	✓	✓	✓
		Mapping controls per interface	✓	✓	
	Supplemental Processing	Convergence control per transfer, per interface	✓	✓	✓
		Ramping per transfer, per interface	✓	✓	✓
		Under-relaxation per transfer, per interface	✓	✓	✓
		Quasi-Newton solution stabilization	✓	✓	✓
Coupled Run Execution	**Participant Execution**	Automated startup/shutdown (including error handling)	✓	✓	✓
		Participant solution sequence control[3 (p. 28)]	✓	✓	
		Simultaneous participant solutions	✓	✓	
		Participant update frequency controls[3 (p. 28)]	✓	✓	
		Participant timeout connection interval	✓	✓	
	System Coupling Execution	Creation/restoration of snapshots of the coupled analysis state	✓	✓	
		Interactive solution commands	✓	✓	
		Reconnection to running System Coupling session	✓		

Category		Capabilities	GUI	CLI	WB
		Execution of System Coupling setup exported from Workbench	✓	✓	
		Project and design point updates via RSM			✓
		Solve mode	✓	✓	
		Mapping mode[5 (p. 28)]	✓	✓	
	Restarts	Coordinated restart point creation	✓	✓	✓
		Restart points written per coupling step	✓	✓	✓
		Restart points written per coupling iteration[3 (p. 28)]	✓	✓	
		Coordinated restart point selection and automated restarts	✓	✓	
	Parallel Processing	Configurable, script-based submission to clusters	✓	✓	
		System Coupling run as distributed processes	✓	✓	✓
		System-defined and custom resource partitioning across participants running in parallel	✓	✓	
		Solver-specific HPC parallel arguments	✓	✓	
Co-Simulation Output	**Charting**	Chart views: • Coupling Step • Coupling Iteration • Simulation Time (for transient analyses)	✓	✓	✓
		Data transfer convergence chart, per simulation	✓		✓
		Data transfer diagnostics chart, per quantity type	✓		
		Export charting data to `.csv` files, per coupling interface	✓	✓	
		Collated participant convergence and monitors			✓
	Transcripts & Log Files	Dynamically generated coupled analysis Transcript and Log file	✓	✓	✓
		Runtime debug output	✓	✓	✓
		Initial mesh, data transfer, and mapping diagnostics	✓	✓	✓
		System Coupling and participant wall-clock time summary	✓	✓	✓

Category		Capabilities	GUI	CLI	WB
		Participant transcript output available for viewing	✓		✓
	Postprocessing	Surface and volume coupling output in EnSight Gold format	✓	✓	
		Live EnSight postprocessing visualization during co-simulation	✓	✓	

1: For more information about Participant Library APIs, see the **Multiphysics** section of the Ansys API Documentation site.

2: This is a beta-level capability. For more information about beta functionality, see the *System Coupling Beta Features* documentation.

3: Use of this capability depends on support by coupling participants. For information on which participants support this feature, see Participant Support of System Coupling Capabilities (p. 29).

4: For more information about supported data transfer quantities, see Participant Variables and Quantity Types Supported by System Coupling (p. 28).

5: Mapping mode and server participants can be used only for one-way transfers of modal data between a Mechanical Server participant and a CFD Server participant in an aerodamping co-simulation.

6: The 2D participant defines x, y, plane coordinates and x, y, vector variables. However, System Coupling uses 3D coordinates internally.

For details about the System Coupling capabilities listed here, see the subsequent sections of this guide.

Participant Variables and Quantity Types Supported by System Coupling

System Coupling supports the use of any vector or scalar variables. For participant-specific information on the variables can be transferred via System Coupling, see the corresponding participant documentation:

- Variables Available for System Coupling in the *Fluent User's Guide*
- Variables Available for System Coupling in the *CFX-Solver Manager User's Guide*
- Variables Available for System Coupling in the *Mechanical User's Guide*
- Variables Available for System Coupling in the *Forte User's Guide*
- Variables Available for System Coupling in the *Maxwell Help*

To facilitate most common coupling workflows, System Coupling also provides **data transfer quantity types** that are referenced by participant variables. Coupling participants declare variables of a given quantity type, and System Coupling ensures that data transfers are created between variables of the same/compatible quantity type.

Available quantity types are listed in the table below.

Table 4: Data transfer quantity types supported by System Coupling

Quantity Type[1] (p. 29)
Force
Incremental Displacement [5] (p. 29)
Heat Rate [6] (p. 29)
Temperature
Heat Transfer Coefficient [7] (p. 29)
Convection Reference Temperature [6] (p. 29)
Mode Shape
Electric Conductivity
1: Quantity types noted in this table are in SI units. **2:** All vector quantities consist of three Cartesian components. **3:** The type of mapping used depends on whether the variable is or extensive (conservative) or intensive (profile-preserving). For more information, see System Coupling's Mapping Capabilities (p. 282). **4:** Transfers of Force Density to Mechanical are not supported for System Coupling in Workbench or cases where Mechanical uses shells and force is being transferred to both sides of any shell. **5:** This is displacement during the current coupling step (that is, since the end of the previous coupling step). **6:** Intensive variables of this quantity type are calculated per-unit. **7:** For a given source/target pair of regions, if a transfer of quantity type Heat Transfer Coefficient is defined, then a transfer of quantity type Convection Reference Temperature must also be defined, and vice versa.

Participant Support of System Coupling Capabilities

The following tables show participant support of the key coupling capabilities available for System Coupling's graphical user interface (**GUI**), System Coupling's command-line interface (**CLI**), and System Coupling in Workbench (**WB**).

Participant-specific support is listed for the following solvers:

- AEDT (Electronics Desktop / Maxwell) (p. 30)
- CFX (p. 31)
- Fluent (p. 31)
- Forte (p. 32)

- Functional Mock-up Unit (FMU) (p. 32)
- Mechanical (p. 33)

Participant-specific support is listed for the following servers:

- CFD Server (p. 33)
- Mechanical Server (p. 34)
- SCDT Server (p. 34)

Table 5: Solver Participant Support of System Coupling Capabilities

Category		Capability	GUI	CLI	WB
AEDT (Electronics Desktop / Maxwell)	**Analysis**	Eddy Current (3D) - Steady and Transient	✓	✓	
		Transient (2D & 3D)	✓	✓	
		Magnetostatic (3D) - Steady and Transient	✓	✓	
	Setup	Define motion on coupled bodies	✓	✓	
		System Coupling time-dependent conditions	✓	✓	
		Time-dependent excitations and/or motion (for Eddy-Current 3D analyses)	✓	✓	
	Transfer Quantities	Volume outputs: • Heat Rate • Lorentz Force (Magnetostatic only)	✓	✓	
		Volume inputs: • Temperature • Electrical Conductivity (Magnetostatic only)	✓	✓	
		Surface outputs: • Heat Rate (2D Transient only) • Heat Rate Density (Magnetostatic only)	✓	✓	
		Surface inputs: • Temperature (2D Transient only)	✓	✓	
	Execution	Participant update frequency controls (for steady coupled analyses)	✓	✓	
		Mesh updates and remapping per step (for System Coupling time-dependent solutions)	✓	✓	

Category		Capability	GUI	CLI	WB
CFX	**Analysis**	Steady	✓	✓	✓
		Transient	✓	✓	✓
	Transfer Quantities	Surface outputs: • Force • Heat Rate • HTC + Tref	✓	✓	✓
		Surface inputs: • Incremental Displacement • Temperature	✓	✓	✓
	Execution	Automatic detection of same interface sides for thin bodies	✓	✓	
Fluent	**Analysis**	Steady	✓	✓	✓
		Transient	✓	✓	✓
	Setup	Customizable solver input files for initial and restarted coupled analysis runs	✓	✓	
		Define motion on coupled bodies	✓	✓	
	Transfer Quantities	Surface outputs: • Force • Heat Rate • HTC + Tref • Temperature • Any custom variable defined via Fluent UDMs	✓	✓	✓
		Surface inputs: • Incremental Displacement • Heat Rate • HTC + Tref • Temperature • Any custom variable defined via Fluent UDMs	✓	✓	✓

Category		Capability	GUI	CLI	WB
		Volume outputs: • Temperature • Electrical Conductivity • Any custom variable defined via Fluent UDMs	✓	✓	
		Volume inputs: • Heat Rate • Lorentz Force • Any custom variable defined via Fluent UDMs	✓	✓	
	Execution	Automatic detection of same interface sides for thin bodies	✓	✓	
Forte	**Analysis**	Steady	✓	✓	
		Transient	✓	✓	
	Transfer Quantities	Surface outputs: • Force • Heat Rate • HTC + Tref	✓	✓	
		Surface inputs: • Incremental Displacement • Temperature	✓	✓	
	Execution	Automatic detection of same interface sides for thin bodies	✓	✓	
Functional Mock-up Unit (FMU)	**Analysis**	Steady	✓	✓	
		Transient	✓	✓	
	Transfer Quantities ,	Anything defined in the FMU FMU participants support single-valued scalar transfer quantities. All noted outputs/inputs are supported for surfaces and volumes involved in a transfer with an FMU region (with `Topology` set to ***`Undefined`***).	✓	✓	

Category		Capability	GUI	CLI	WB
Mechanical	**Analysis**	Steady	✓	✓	✓
		Transient	✓	✓	✓
	Setup	Customizable solver input files for restarted coupled analysis runs One region per interface side only	✓	✓	
	Transfer Quantities	Surface outputs: • Force • Heat Rate • Incremental Displacement • HTC + Tref • Temperature	✓	✓	✓
		Surface inputs: • Heat Rate • Incremental Displacement • HTC + Tref • Temperature	✓	✓	✓
		Volume outputs: • Temperature	✓	✓	
		Volume inputs: • Heat Rate • Temperature	✓	✓	
	Execution	Automatic detection of same interface sides for thin bodies	✓	✓	

Table 6: Server Participant Support of System Coupling Capabilities

Category		Capability	GUI	CLI
CFD Server	**Analysis**	Steady	✓	✓
	Setup	Add aerodamping data transfer groups	✓	✓
		Check for changes to participant input file	✓	✓
		Update participant after input file modifications	✓	✓

Category		Capability	GUI	CLI
		Participant instancing	✓	✓
	Transfer Quantities	Surface inputs: • Mode Shape	✓	✓
	File Format	CSV format for mode shape imports into the blade flutter analysis (CFX or Fluent)	✓	✓
Mechanical Server	**Analysis**	Steady	✓	✓
	Setup	Add aerodamping data transfer groups	✓	✓
		Multiple regions per interface side	✓	✓
		Check for changes to participant input file	✓	✓
		Update participant after input file modifications	✓	✓
	Transfer Quantities	Surface outputs: • Mode Shape	✓	✓
	File Format	Mechanical RST format as source for mode shapes from Mechanical Modal analysis	✓	✓
SCDT Server	**Analysis**	Steady	✓	✓
		Transient Subject to certain limitations as documented in the `SCDTFileFormatSPecification.md` file.	✓	✓
	Transfer Quantities	All supported quantities	✓	✓
	File Format	SCDT file format (as documented in `SCDTFileFormatSPecification.md`)	✓	✓

For information on a given participant's support for System Coupling capabilities, see its coupling-related product documentation:

- "System Coupling Analyses Using Maxwell" in the *Maxwell Help*
- Coupling CFX to an External Solver: System Coupling Simulations in the *CFX-Solver Manager User's Guide*
- Performing System Coupling Simulations Using Fluent in the *Fluent User's Guide*
- System Coupling in the *Forte User's Guide*
- System Coupling in the *Mechanical User's Guide*
- Ansys Rocky and Ansys Mechanical in the *Rocky User Manual*

- The `SCDTFileFormatSPecification.md` Markdown file available in `<Installation Path>\v251\SystemCoupling\Participants\ScdtServer`. You can view this file with any Markdown-compatible reader.

System Coupling Contexts: GUI, CLI, and Workbench

You can run System Coupling in the following contexts:

System Coupling's User Interfaces:

- ***System Coupling's Graphical User Interface (GUI)***

  Set up and execute coupled simulations using the **System Coupling** GUI. A robust API provides access to System Coupling's infrastructure and functionality, enabling you to create and solve multiphysics simulations involving the transfer of surface and volume data between coupling participants.

- ***System Coupling's Command-Line Interface (CLI)***

  Set up and execute coupled simulations using the System Coupling CLI. System Coupling offers the same capabilities when run from its CLI as it does when run from its GUI.

For more information, see Using System Coupling's User Interfaces (p. 53).

System Coupling in Workbench

Workbench's **System Coupling** component system is an easy-to-use, all-purpose infrastructure that enables you to build and run simulations involving the transfer of surface data between coupling participants.

- **Setup**

  Use the Workbench interface and workflow to create coupled analyses. Coupled analyses that were set up in Workbench can either be run in Workbench or exported for execution in System Coupling's GUI or CLI.

- **Execution**

  Use the Workbench interface and workflow to run coupled analyses that were set up in Workbench.

For more information, see Using System Coupling in Workbench (p. 237).

Steps of a System Coupling Analysis

Coupled analyses performed using System Coupling are composed of the following general steps:

1. **Set up the physics for coupling participants.**

   For each participant, set up data transfer variables and regions to be coupled, as well as coupling-related settings that enable a coupled analysis.

2. **Set up System Coupling.**

Set up the System Coupling part of the analysis, specifying analysis settings and defining data transfers.

3. **Run the coupled analysis.**

   Run the coupled analysis by starting System Coupling and each of the participant solvers.

4. During the coupled analysis run, you can:

   - **Monitor solution progress.**
   - **Stop and resume the analysis.**

5. After the coupled analysis run, you can:

   - **Extend and restart the analysis.**
   - **Debug your coupled analysis.**

6. **Review the analysis output.**

   Review the solution output produced by both System Coupling and participant solvers. You can postprocess the results using the methods recommended for a given participant or by loading coupling results into Ansys EnSight.

System Coupling Analysis Types

When using System Coupling, you can perform the following types of analyses:

Steady/General

- Used when all coupling participants are performing a steady-state or static solution.
- Accurate coupled solutions can be achieved using different combinations of coupling steps and coupling iterations.
- Depending on the analysis setup, there may be no steps (iterations only), one step, or multiple steps.

For more information, see Steady Analysis Type (p. 37).

Transient

- Used when any coupling participant is performing a transient solution.

> **Note:**
>
> For transient solutions involving physics with disparate time scales, System Coupling offers mixed ***steady-transient*** analyses. For more information, see Steady-Transient Analyses (p. 38).

- A coupling step is associated with a time interval when the coupling step size (in seconds) is specified. With a time specified, a coupling step is the same as a time step within the transient analysis.

For more information, see Transient Analysis Type (p. 38).

Steady Analysis Type

This section describes the ways coupling steps and coupling iterations can be combined in a steady coupled analysis (that is, an analysis that has **Analysis Type** set to **Steady** if set up in the GUI or CLI, or to **General** if set up in Workbench). In a steady analysis, the specific combination of coupling steps and coupling iterations:

- determines when restart data can be written,
- allows you to balance the required file storage space and the need for analysis restarts, and
- determines how you can use System Coupling's under-relaxation factor and ramping, as these only apply to coupling iterations and cannot be applied over coupling steps.

Steady coupled analyses are generally ***iteration-based***. An iteration-based analysis can have multiple coupling iterations. Multiple coupling steps cannot be defined.

When an iteration-based coupled analysis is solved, the analysis continues executing until either the solution converges, or the specified maximum number of coupling iterations is completed.

Restart points are written at the end of coupling iterations according to the iteration-based **Output Control** settings in the data model. The creation of intermediate restart points helps to ensure restart coverage, so that restarts are possible even when a simulation with a lengthy runtime is terminated either abnormally (for example, due to an error or power interruption) or via an *Abort* operation. (An *Interrupt* operation allows the current coupling iteration to finish, creates a restart point, and then cleanly stops the coupled analysis.)

For any coupled analysis, the creation of restart points is a trade-off to be weighed against available resources. Minimizing the number of restart points created minimizes file storage space at the expense of the ability to restart the analysis, while maximizing the number of restart points can save time and computational effort if a restart is required, but at the expense of file storage space.

Ramping and under-relaxation can be applied across coupling iterations.

Note:

The exception is the steady analysis which is ***step-based*** because it includes a participant that can create restart points only for coupling steps — that is, with a **Participant Type** of **AEDT** (only for 2D participants), **CFX**, **MAPDL**, or **EXTERNAL DATA**). This type of analysis can have one or more coupling steps, with multiple iterations permitted for each.

When a step-based coupled analysis is solved, the analysis continues executing until the specified number of coupling steps is completed. The transition from one coupling step to the next occurs when either the solution converges, or the specified maximum number of coupling iterations is completed.

Restart points are written only at the end of coupling steps, according to the step-based **Output Control** settings in the data model. This limits restart capabilities, particularly for single-step analyses (for which a restart point is written only at the end of a solution) or particularly complex or long-running multi-step analyses which might experience abnormal terminations.

Ramping and under-relaxation can be applied across coupling iterations, but not across coupling steps. For multi-step steady analyses, the full data transfer value is transferred at the end of each coupling step. Participant solvers may ramp data received from System Coupling at the coupling steps.

For best practices, see Improving the Accuracy of Steady Analyses (p. 398).

Transient Analysis Type

In a transient analysis, a coupling step is associated with a time interval when the coupling step size (in seconds) is specified. When running a transient analysis, values must be defined for the end time and the time step size. For more information, see `EndTime` and `TimeStepSize` in the *System Coupling Settings and Commands Reference* manual.

With a time specified, a coupling step is the same as a time step within the transient analysis. The coupling step size used should reflect the time scales of the physics being studied. Note that unless sub-stepping is supported by the coupling participants, coupling step size is typically limited by the finest/smallest time scale of the participants.

If the analysis duration is specified using an end time, ensure that an integral number of coupling steps can be executed between the (re)start time and the specified end time. If this is not done, then the final coupling step size is reduced to respect the specified end time, and this may introduce temporal discretization error into the coupled analysis.

The minimum number of coupling iterations may be set to a value larger than one (one is the default). If the data transfers have been under-relaxed, ensure that a minimum number of coupling iterations is performed so that you iterate out the effect of the under-relaxation. Note, however, that the data transfer convergence criteria would usually make this unnecessary. The maximum number of coupling iterations should be set to allow complete convergence within each coupling step.

Steady-Transient Analyses

Mixed ***steady-transient*** solutions (in which one participant runs a steady solution and one runs a transient solution) have a Transient analysis type. By mixing steady and transient participant analyses, greater solution efficiency can be realized for transient analyses involving physics with disparate time scales. The coupled analysis remains time-accurate, provided that the physical timescales of the steady-state participant(s) are orders of magnitude faster than those of the transient participant(s).

In such steady-transient analyses, System Coupling will show the overall analysis type as **Transient**, while the physics with the shorter/faster timescale is solved as steady-state.

Note:

For steady-transient analyses with an AEDT coupling participant:

- When the AEDT participant has excitations or motion that is dependent on System Coupling time, its solution type is presented to System Coupling as Transient, even if the underlying electromagnetic solution type is Eddy Current.
- You can further improve solution efficiency by moderating the update frequency for the AEDT participant. For more information, see `UpdateControl` in the *System Coupling Settings and Commands Reference* manual.

For best practices, see Improving the Accuracy of Transient Analyses (p. 400).

System Coupling Settings

All coupled analyses have settings that enable you to control various aspects of the analysis. The following types of settings are available:

- ***Library settings***

  For System Coupling's user interfaces, define global expression, and transformation objects that can be referenced elsewhere in the analysis.

- ***Coupling Participant settings***

  For each participant, define a name, regions from and to which data can be transferred, the input and output data transfer variables available for each region, the update frequency, and execution controls.

- ***Analysis Control settings***

  Define the type of analysis and controls for the analysis type, partitioning algorithm, simultaneous participant updates, and global stabilization.

- ***Coupling Interface settings***

  Define coupling interfaces, including data transfer details and mapping controls.

- ***Solution Control settings***

  Define solution-level details such as the solution duration, time step size, and number of coupling iterations per step.

- ***Output Control settings***

  Control the frequency with which coupling output is generated.

Units in System Coupling

System Coupling can perform unit conversions on certain entries in the data model, such as time step size. In addition, System Coupling can perform unit conversions for MAPDL participants. All other participants are expected to provide SI units.

For details on the usage of units in System Coupling, see the following sections:

Unit Conversions

System Coupling uses the SI unit system internally and accepts many common units.

If no units are specified, then System Coupling assumes that SI units are being used.

If non-SI units are specified, then System Coupling converts them to SI units for internal processing and back to the participant unit system for consumption by the coupling participant.

Unit Syntax

Dimensional quantities are defined in units or a combination of units. The unit syntax used in System Coupling is defined as follows:

```
[multiplier|unit|^power]
```

where:

- **multiplier** is a multiplying quantity (for example, **p** for **pico**, **c** for **centi**, **M** for **mega**)
- **unit** is the unit string (for example, **g**, **m**, **J**)
- **power** is the power to which the unit is raised

Specifying Units in System Coupling

When units are specified in System Coupling's data model, the following rules apply:

- When units are typed in, they must be enclosed by square brackets: **[...]**
- When units are selected from a list, the brackets are typically not visible, but are assumed.
- When no units are specified, System Coupling defaults to the SI unit system.

Declaring Units in System Coupling

Units declarations must obey the following rules:

Multipliers

- Short forms of the multiplier are usually used. For details, see Unit Multipliers (p. 41).

Units

- A unit string consists of one or more units quantities, each with an optional multiplier and optional power.
- Each separate unit quantity is separated by one or more spaces.
- Unit strings are case sensitive. For example, **Kg** and **KG** are both invalid as parts of unit strings.

Powers

- Powers are denoted by the caret symbol (**^**).
- If no power is specified, a power of 1 is assumed.
- The **/** operator is not supported, so a negative power is used for unit division. For example, **[kg m^-3]** corresponds to kilograms per cubic meter.

Note:

- If you enter units that are inconsistent with the physical quantity being described, then a dialog box will appear informing you of the error and the units will revert to the previous units.
- Units do not have to be given in terms of the fundamental base units (mass, length, time, temperature, angle, and solid angle). For instance, **Pa** (Pascals) and **J** (Joules) are both acceptable as parts of unit strings.

Unit Multipliers

System Coupling supports the unit multipliers shown in the table below.

Table 7: Unit multipliers supported by System Coupling

Long Form	Short Form	Factor	Notation
exa	**E**	10^{18}	1E+18
peta	**P**	10^{15}	1E+15
tera	**T**	10^{12}	1E+12
giga	**G**	10_{9}	1E+09
mega	**M**	10^{6}	1E+06
kilo	**k**	10^{3}	1000
hecto	**h**	10^{2}	100
deca, **deka**	**da**	10^{1}	10
deci	**d**	10^{-1}	0.1
centi	**c**	10^{-2}	0.01

Long Form	Short Form	Factor	Notation
milli	**m**	10^{-3}	0.001
micro	**u**	10^{-6}	1E-06
nano	**n**	10^{-9}	1E-09
pico	**p**	10^{-12}	1E-12
femto	**f**	10^{-15}	1E-15
atto	**a**	10^{-18}	1E-18

Commonly Used SI Units

The following sections list SI units that are commonly used in System Coupling:

SI Base Units

If units are not specified, System Coupling uses a system that is similar to the International System of Units (SI), except that angles are measured in radians. ***Base units*** are units are used as the foundation for all other units. The table below lists the SI base units supported by System Coupling.

Table 8: SI base units

Quantity	Long Form	Short Form
Angle[1] (p. 42)	**radian**	**rad**
Amount of Substance	**mole**	**mol**
Current	**ampere**	**A**
Length	**meter**	**m**
Luminous Intensity	**candela**	**cd**
Mass	**kilogram**	**kg**
Time	**second**	**s**
1: Although defined as a base unit in SystemCoupling, **radian** is technically a derived unit. See SI Derived Units (p. 43).		

SI Data Transfer Quantity Units

The table below shows the SI units System Coupling uses for data transfer quantities.

Table 9: SI data transfer quantity units

Quantity	Variable Type	Long Form	Short Form
Convection Reference Temperature	Intensive	kelvin	K
Force	Extensive	newton	N
Incremental Displacement	Intensive	meter	m
Heat Rate	Extensive	watt	W
	Intensive	watts per cubic meter	W/m^3
Heat Transfer Coefficient	Intensive	kilograms per cubic meter	kg s^-3 K^-1
Temperature	Intensive	kelvin	K

SI Derived Units

The table below shows a subset of the SI derived units supported by System Coupling.

Table 10: SI derived units supported by System Coupling

Quantity	Long Form	Short Form	Base Unit Derivation
Temperature	degrees Celsius	C	K - 273.15
Energy	joule	J	N·m
Force	newton	N	kg·(m/s^2)
Frequency	hertz	Hz	/s
Power	watt	W	J/s
Pressure	pascal	Pa	N/m^2
Plane Angle	radian	rad	m·m-1

Commonly Used Non-SI Units

The table below shows a subset of commonly used non-SI units supported by System Coupling.

Table 11: Non-SI units

Quantity	Long Form	Short Form
Angle	degree	deg
Amount of Substance	pound mole	lbmol
	slug mole	slugmol
Current	ampere	A

Quantity	Long Form	Short Form
Length	inch	in
	foot	ft
	yard	yd
	mile	mi
Mass	tonne	t
	pound mass	lbm
	pound	lb
	ounce	oz
	ton	t
	slug	lbm/32.2
	slinch	lbm/386.4
Pressure	pound-force per square inch	psi lbf/in^2
Time	minute	min
	hour	hr
Temperature	Celsius	C
	Fahrenheit	F
	Rankine	R

Stops and Restarts of System Coupling Analyses

In some cases, you may want to stop a coupled analysis during its execution. If you've interrupted the run, you can restart the analysis from a specified restart point.

For general information on stops and restarts, see:

Stopping a Coupled Analysis

Restarting a Coupled Analysis

Stopping a Coupled Analysis

You can stop a coupled analysis run by either interrupting or aborting it.

- An ***interrupted*** analysis produces a clean stop, where the run continues until the current coupling step is finished and the restart data are generated. An interrupted run can be restarted later from end of the coupling step in which it was stopped.
- An ***aborted*** analysis terminates the run immediately and does not produce restart data. An aborted run cannot be restarted from the coupling step in which it was stopped. However, an aborted can be restarted from an intermediate restart point that was completed before the analysis was aborted.

For context-specific details, see:

- Stopping a Coupled Analysis (p. 141)
- Stopping a System Coupling Analysis in Workbench (p. 267)

Restarting a Coupled Analysis

You can restart a completed or interrupted coupled analysis run, either from the final restart point or from an intermediate restart point. Restarts require relevant files from System Coupling and each of the coupling participants.

Note:

Some participants may not allow restarts. For example, Mechanical participants do not support restarts for steady-state thermal and transient thermal analyses.

If any coupling participant does not support restarts, then restarts are disabled for the coupled analysis. A participant's restart capabilities may vary according to the type of analysis being performed.

To restart an interrupted analysis, perform the following steps:

1. **Generate restart files.**

   Information needed to restart the System Coupling is contained in the Restart files (p. 351) generated by System Coupling during the run.

   Information needed to restart the coupling participants, as well as the act of restarting those participants, are managed by the participants themselves.

   Tip:

   To ensure that participant solvers are writing restart data at synchronized coupling steps, the creation of restart points is controlled by System Coupling. For details on how a given participant handles restarts, use the links provided in Supported Coupling Participants for System Coupling in Workbench (p. 238) or Supported Coupling Participants (p. 54) to access its documentation.

   ---

2. **Execute the restart run.**

   Once the coupled analysis run is finished or interrupted, or if the solution fails, you can restart the run from any of the saved restart points. The same restart point must be used by System Coupling and all coupling participants.

   If you've saved changes to analysis settings for System Coupling and/or any participant except for Mechanical, then the changes are written to a Settings file (p. 351) and included in the restarted run.

Details for the restarted run are appended to the end of the existing `scLog.scl` file.

Note:

Convergence history for a restarted run is generally not identical to that observed in a continuous run. The following factors contribute to changes in convergence:

- Participants are not guaranteed to deliver identical behavior.
- Interfaces are remapped upon restart, potentially changing the interpolation weights.
- Changes in convergence behavior do not produce different results, provided that solutions and data transfers are fully converged within the coupling steps.

For context-specific information on restarts, see:

- Restarting a Coupled Analysis (p. 142)
- Restarting a Coupled Analysis in Workbench (p. 268)

Migrating to Ansys System Coupling 2025 R1

As improvements are made to System Coupling's data model and functionality, care is taken to maintain backward compatibility. With each new release, however, Ansys recommends that you familiarize yourself with any changes and identify areas where your scripts or workflows may be affected.

This section provides you with the information necessary for a smooth transition of System Coupling to the current release, **Ansys 2025 R1**. The following sections describe changes that have been implemented since the previous release and which may impact your usage of System Coupling:

Migrating Across Multiple Releases

If you are migrating across multiple releases, you may find it helpful to consult the migration instructions for the intervening releases, as well. Starting with the 2019 R2 release, migration guidance for command-line System Coupling has been published in the *System Coupling User's Guide*. To access this documentation:

1. Go to the **System Coupling** page of the online **Ansys Help** site.
2. At the upper-right corner of the window, select the appropriate release from the **Release** menu.
3. Under **Documentation**, open the *System Coupling User's Guide*.
4. Use the search field to locate the "Migrating Command-Line System Coupling Scripts" section.

Migrating Command-Line System Coupling Scripts

System Coupling 2025 R1 contains changes that can impact existing coupling scripts.

Existing coupling cases (that is, that have been saved to a `Settings.h5` file) will be migrated automatically when you re-open them with the most recent release of System Coupling. If you have existing coupling scripts, however, you may need to update them so they will continue to work with the current and later releases.

The following sections contain information on changes introduced in the 2025 R1 release which may affect existing scripts, as well as instructions on how to migrate scripts when necessary:

Changes to the Data Model

The following table lists changes to the data model in the 2025 R1 release of System Coupling.

Data Model Item	Change	Recommendation
`CouplingParticipant.Variable.Attribute.Modifiable`	**New setting**	See Modifiable
`Library.Instancing.RotationAxis`	**New setting**	See RotationAxis
`Library.Instancing.Axis`	**New setting**	See Axis
`Library.Instancing.RotationalOffset`	**New setting**	See RotationalOffset
`Library.Instancing.AxisFrom`	**New setting**	See AxisFrom
`Library.Instancing.AxisTo`	**New setting**	See AxisTo
`OutputControl.LiveVisualization.WriteResults`	**Moved to beta feature**	See WriteResults

Changes to Commands

The following table lists changes to commands in the 2025 R1 release of System Coupling.

Command	Change	Recommendation
`UpdateParticipant`	**AEDT participant now supported**	For details, see `UpdateParticipant`.

Known Issues and Limitations

This section lists issues and limitations that are known to exist in Ansys System Coupling 2025 R1. When available, workarounds are provided.

Issues and limitations exist in the following categories:

Parallel Processing

Distributed parallel processing is not supported for Windows

Distributed parallel processing is not supported for Windows systems. As a result, the `--cnf` command-line argument is not available for Windows and, if issued, will cause System Coupling to generate an exception.

For local jobs that were set up using the `--cnf` argument, you may instead use the `-t` option to specify the number of cores.

For more information on configuring parallel participant execution on Windows, see Parallel Processing for Coupling Participants (p. 166).

Error for RSM job submission using Microsoft HPC Pack 2019

When submitting a job to RSM on a machine that uses Microsoft HPC Pack 2019, you may receive an error message indicating that the coupling process failed to start.

To address this issue, submit the job on another resource, either by using an earlier version of HPC Pack or by submitting the job on a Linux system.

When submitting a job to RSM on a machine that uses Microsoft HPC Pack 2019, you may receive an error message indicating that the coupling process failed to start.

To address this issue, submit the job on another resource, either by using an earlier version of HPC Pack or by submitting the job on a Linux system.

For RSM jobs, CFX participants must be manually configured to run in the foreground

System Coupling jobs sent to RSM must run in the foreground, and all coupling participant setups must be consistent with that of System Coupling. Any coupling participant with its **Solution** cell's **Solution Process | Update Option** property set to **Run in Background** will fail to connect to System Coupling.

To avoid this issue, ensure that any CFX coupling participant has its **Solution | Update Option** property set to **Run in Foreground** before submitting a coupling job to RSM.

Once an update has been attempted with a CFX participant's **Solution | Update Option** property set to **Run in Background**, you may find that it is not possible to change the value of this setting for CFX. To address this issue, first reset System Coupling's **Solution** cell and then set CFX to perform updates using **Run in Foreground**.

Units

A small subset of unit multipliers and Length unit strings cannot be used

Currently, the following unit multiplier cannot be used:

- Short form **da** (use **deca** or **deka** instead)

Currently, the following Length unit strings cannot be used:

- Long form **meter** (use **m** instead)
- Short form **mi** (use **mile** instead)

Graphical User Interface

Detached instances of System Coupling

A System Coupling session that was started in GUI Server mode, as described in Reconnecting the GUI to a Running Coupled Solution (p. 139), can be ended cleanly only from a connected instance of the **System Coupling** GUI. If this is not possible, you must end it manually by issuing an OS-specific kill request.

To do this, you will need the process ID for the job to be killed. This information is available in Transcript/Log for the coupled analysis.

Orphaned GUI Server files

In most cases, the GUI Server (`sycGui.srv`) file (p. 331) should not be removed manually. This file is automatically removed from the coupling working directory the instance of System Coupling running in GUI Server mode exits. However, unexpected termination of a coupling run may leave orphaned (that is, not connected to any active session) GUI Server files, which prevents subsequent instances of System Coupling from starting in GUI Server mode in that directory.

If a previous run terminated unexpectedly or was killed manually (or if you are otherwise certain that no System Coupling instance is running in GUI Server mode in that directory), then you can delete the orphaned file and restart System Coupling.

Orphaned Solution Lock files

In most cases, the Solution Lock (`sycSolve.lock`) file (p. 331) should not be removed manually. This file is automatically removed from the coupling working directory when the solution stops. However, unexpected termination of a coupling run may leave orphaned (that is, not connected to any active session) Solution Lock files, which blocks any subsequent solves in that directory.

If a previous run terminated unexpectedly or was killed manually (or if you are otherwise certain that no solution is running in that directory), then you can delete the orphaned file and reattempt a solve.

GUI scaling issues on 4K monitors

Users of the **System Coupling** GUI with 4K monitors may encounter incorrect scaling of text and other components of the GUI. If this happens, set the following environment variables:

```
QT_AUTO_SCREEN_SCALE_FACTOR = 0
QT_SCALE_FACTOR = 2
```

The value of ***2*** for the QT_SCALE_FACTOR variable is suggested, but some experimentation with different values might be required.

For details, see Environment Variable Reference in the *QT Documentation*.

Non-automatic proxy settings may prevent the GUI from reconnecting

In some cases, non-automatic proxy settings on your computer may prevent the **System Coupling** GUI from reconnecting to a running coupled solution. When this occurs, you will receive a message indicating the nature of this error.

Note:

Currently, this issue has been reported only on Windows systems, but could potentially occur on a Linux system, as well.

You may address the issue using the following workarounds:

- **Workaround 1: Switch to an automatic proxy setting.**
  - **Windows:** Go to **Settings > Network & Internet > Proxy** and ensure that **Automatically detect settings** is turned to ***On***.
  - **Linux:** If the error occurs on a Linux system, contact your IT support team for instructions.
- **Workaround 2: Use an environment variable to disable proxying.**

  Use this workaround if the issue was not resolved by using an automatic proxy setting OR when your machine is required to use a non-automatic setting.

  Create a new system environment variable called SYC_NO_PROXY. No value is required.

GUI issues caused by inadequate remote graphics setup

When the **System Coupling** GUI is run on a remote system without an adequate remote graphics display set up it may crash at start-up. (For details on graphics setup requirements, see the related Ansys installation documentation.)

If issues are encountered, try setting the environment variable QT_QUICK_BACKEND to ***software*** before starting the GUI.

For details, see Switch Between Adaptations in Your Application in the *QT Documentation*.

Charting Limitation

If a single iteration is performed per time step, then some or all of the charting data may be plotted on the wrong time step. This is similarly true for the charting data file.

Participants

Participant connection issues

A participant may fail to connect to System Coupling if it attempts to use the wrong host name for the machine on which System Coupling is running. This can occur if the hosts file on that system has been modified manually (for example, to circumvent access to network locations).

To resolve this, ensure that the hosts file does not contain arbitrary entries. The hosts file is typically located in the **/etc** directory on Linux platforms and **C:\Windows\System32\Drivers\etc** on Windows platforms.

Modal Analysis

Modal analysis is not supported if the Mechanical `*.rst` file contains non-complex mode-shape data and **Damped** is enabled in the **Analysis Settings** of the Mechanical setup.

Data Model

Data Model Validation Errors

In release 2024 R1 and later, automatic defaulting for the **Coupling Participant** and **Region List** data model values has been removed. For cases saved in release 2022 R1 and earlier, these extended values were not saved. When you open the case in the latest release the values are missing, causing validation errors. To fix this issue, you must manually set the values and re-save the case.

Changes to AEDT Case Setups

Changes to AEDT case setups may require you to regenerate SCP files. You can update an existing AEDT participant setup in System Coupling using the `UpdateParticipant` command with the updated SCP file.

Postprocessing in EnSight

Mapping Statistics Variables Name Change

A change to how mapping statistics variables are named (p. 216) has caused a change in the name of the variable data files. This can cause a mix of file names with the old and new naming when restarting a case from an intermediate result. To work around this issue, re-run the initial case and generate new EnSight results files.

Live Visualization Connection Issue

In a live visualization object, when the **Show Viewer** setting is *False*, the EnSight user interface should be launched. However, there may be a connection issue when using VPN, or a network misconfiguration. To solve this issue, enable beta features and under **Solution Control**, set **Use Local Host When Possible** to *True* to force using `127.0.0.1` as the host name for EnSight. Note that **Use Local Host When Possible** is a global setting that also affects the connection to participant solvers, so will also force the host name to the solvers.

Using System Coupling's User Interfaces

When running System Coupling in one of its user interfaces, you can use the graphical user interface (GUI) and/or the command-line interface (CLI) to build and execute coupled analyses. The same functionality is available in both interfaces.

Note:

The *System Coupling User's Guide* generally uses context designations as needed to enhance clarity. In this section, however, when the term "System Coupling" is used without qualification, it refers to System Coupling when run from one of its user interfaces.

The following topics are addressed in this section:

- Overview of System Coupling's User Interfaces (p. 54) provides a high-level introduction to using System Coupling in one of its user interfaces, including information on supported participants, coupling interfaces, and data transfers.
- Working with System Coupling's Data Model (p. 82) includes a description of the data model structure, an overview of the root-level containers, and instructions on viewing data model contents.
- Using System Coupling's Graphical User Interface (GUI) (p. 87) provides instructions on starting the GUI and describes its components and available functions.
- Using the System Coupling Command-Line Interface (CLI) (p. 113) provides instructions on starting the CLI and includes information on accessing help and navigating the data model from the command line.
- Preparing for a Coupled Analysis (p. 115) provides instructions on preparing a working directory and setting up participant physics.
- Creating a Coupled Analysis (p. 120) provides instructions on adding participants, interfaces, and data transfers to a coupled analysis.
- Modifying Coupled Analysis Settings (p. 130) provides instructions on changing data model settings and expert settings, as well as on requesting debug output.
- Running a Coupled Analysis (p. 131) provides instructions for running and monitoring a coupled analysis.
- Common Coupled Analysis Tasks (p. 134) describes tasks that are commonly used in coupled simulations, including restarts, snapshots, opening a new working directory, and connecting/reconnecting to running simulations.
- Advanced Coupled Analysis Tasks (p. 148) describes advanced tasks that can be used in coupled simulations, including the use of debugging and diagnostic output, participant timeout connection intervals, geometry transformations, interactive solve commands, simultaneous participant solutions,

parallel processing for HPC job submission, and execution of coupling setups exported from Workbench.

- Reviewing Coupled Analysis Output and Results (p. 197) provides guidance on accessing and working with coupling output, including convergence charts and Ansys EnSight-formatted results.

Overview of System Coupling's User Interfaces

This section provides a high-level overview of how to use System Coupling in its user interfaces. The following topics are addressed:

- Supported Coupling Participants (p. 54) specifies the systems that can serve as participants in a coupled analysis.
- Coupling Interfaces and Data Transfers (p. 56) describes the rules for creating coupling interfaces and data transfers.
- Steps of a Coupled Analysis (p. 58) lists the steps involved in setting up and running a coupled analysis in System Coupling's GUI or CLI.
- Expressions in System Coupling (p. 59) provides instructions on how to use expressions to specify values for data model settings.
- Participant Solution Sequencing (p. 81) describes the process System Coupling uses to determine the order in which participant solutions are executed.

Supported Coupling Participants

System Coupling *participants* are applications (that is, solvers) that send and/or receive data in a coupled analysis. The following is the list of applications that can act as coupling participants in coupled analyses run in one of System Coupling's user interfaces:

CFX

For information on using System Coupling with CFX, see the following sections in the *CFX-Solver Modeling Guide*:

- Coupling CFX to an External Solver: System Coupling Simulations
- System Coupling Related Settings in CFX
- Running CFX as a Participant from System Coupling's GUI or CLI

CFD Server

For information on using System Coupling with a CFD Server participant, see Aerodynamic Damping Analysis (p. 377).

Fluent

For information on using System Coupling with Fluent, see the following sections in the *Fluent User's Guide*:

- Performing System Coupling Simulations Using Fluent
- System Coupling Related Settings in Fluent
- Running Fluent as a Participant from System Coupling's GUI or CLI

Forte

For information on using System Coupling with Forte, see the following sections in the *Forte User's Guide*:

- System Coupling
- System Coupling Related Settings
- How to Set Up and Run a Coupled Simulation

Functional Mock-Up Unit (FMU)

For information on using System Coupling with an FMU, see Functional Mock-Up Unit (FMU) Co-Simulation Participants (p. 357).

Maxwell

For information on using System Coupling with Maxwell, see the following section in the *Maxwell Help*:

- System Coupling Analyses Using Maxwell

Note:

To enable coupling for Maxwell on a Linux system, you must set the **ANSYSEM_ROOT251** environment variable to the location of your Ansys EM Suite installation.

Mechanical

For information on using System Coupling with Mechanical, see the following sections in the *Mechanical User's Guide*:

- System Coupling
- System Coupling Related Settings in Mechanical
- Running Mechanical as a Coupling Participant in System Coupling's GUI or CLI

Mechanical Server

For information on using System Coupling with a Mechanical Server participant, see Aerodynamic Damping Analysis (p. 377).

SCDT Server

For information on using System Coupling with a SCDT Server participant, see the `SCDTFile-FormatSPecification.md` Markdown file available in `<Installation Path>\v251\SystemCoupling\Participants\ScdtServer`. You can view this file with any Markdown-compatible reader.

Coupling Interfaces and Data Transfers

In a coupled analysis performed in one of System Coupling's user interfaces, data are transferred between the two sides of a given coupling interface.

Coupling Interfaces

A ***coupling interface*** defines two sets of regions, each set belonging to one participant, between which data can be transferred during the coupled analysis. A single coupling interface encompasses all transfers in either direction between a given set of regions on the two sides of the interface.

Interfaces support two-way data transfers, so a given participant can be both a source and a target in the same interface. Because of this, both the interface sides and their associated participants and regions are referred to as "side one" and "side two," instead of "source" and "target."

For more information, see Rules for Adding Coupling Interfaces (p. 56).

Data Transfers

A ***data transfer*** is the transfer of a single quantity type in one direction between the two sides of the interface.

Because data transfers go in only one direction, associated participants and regions may still be referred to as "source" and "target" when discussed in relation to a data transfer.

For more information, see Rules for Adding Data Transfers (p. 57).

For more information on supported data transfers, see Participant Variables and Quantity Types Supported by System Coupling (p. 28).

Rules for Adding Coupling Interfaces

When you add a coupling interface (p. 120) in one of System Coupling's user interfaces, the following rules apply:

- Interfaces are given default names according to the convention `Interface-<#>`, where "#" indicates the order in which the interfaces were created. For example, if three interfaces are created for an analysis, they are named `Interface-1`, `Interface-2`, and `Interface-3`.
- Interface names must be unique within an analysis.
- An interface encompasses all data transfers between sets of regions on side `One` and side `Two` of the interface.

- Each side of an interface must be associated with a unique participant and a unique set of regions. The same participant cannot serve as both side One and side Two of the same interface, and only one interface can have a given combination of side One and side Two regions.
- Each interface side can have regions of only a single topology.
- An interface can be created between participant regions with different topologies (for example, planar surface and volume).
  - For data transfers between like topologies, both sides of the resulting interface must have regions of the same topology. For example, if side One of the interface has surface regions, then side Two of the interface must also have surface regions.
  - For data transfers between unlike topologies, each side of the resulting interface can have regions of a different topology. For example, side One can have volume regions and side Two can have planar surface regions.
- Whether a coupling interface can comprise single or multiple source regions depends on participant support of multi-region interfaces.
  - Generally, an interface can include only a single source region and a single target region.
  - The exception is when the interface has a side associated with a Fluent and/or a Maxwell participant. These participants support multiple regions on both source and target locations.
- For optimal mapping on an interface between regions of different topologies, the geometries must be aligned. See Defining Geometry Transformations for Models with Different Orientations (p. 148).
- An interface must have at least one valid data transfer. See Rules for Adding Data Transfers (p. 57) and System Coupling Data Transfers (p. 281).
- In a single interface, there can be multiple data transfers of different quantities in both directions.

Rules for Adding Data Transfers

When you add a data transfer (p. 120) in one of System Coupling's user interfaces, the following rules apply:

- Data transfer objects are assigned default names according to the convention <*TargetVariableDisplayName*>-<*#*>, where "<*TargetVariableDisplayName*>" is the target participant's internal name for the target variable and "*#*" indicates the order in which the data transfers with the same target variable are created on the interface.
- Data transfer names must be unique per coupling interface. For example, if three temperature data transfers with the same target participant are created on three different interfaces, they will be created with the same name. They must be in ASCII format and cannot include forward slashes (*/*).
- Whether a single target region can receive send/data from multiple source regions on an interface depends on the source participant's support of multi-region coupling interfaces. Currently, only Fluent and Maxwell support multiple regions on both source and target locations.

For data transfer creation details that are generally applicable to all System Coupling contexts, see Rules for the Creation of Data Transfers (p. 281).

Steps of a Coupled Analysis

To create and run a coupled analysis in one of System Coupling's user interfaces, perform the following steps:

1. **Prepare for the analysis.** (p. 115)
   a. Create a directory structure. (p. 116)
   b. Set up participants. (p. 117)
   c. Generate participant input files. (p. 119)
2. **Create the analysis.** (p. 120)
   a. Add the participants. (p. 121)
   b. Add the coupling interface(s) and data transfers. (p. 124)
3. **Modify the analysis.** Optionally, you can:
   a. Change data model settings. (p. 130)
4. **Run the analysis.** (p. 131)
   a. Start the solution. (p. 132)
   b. Monitor the solution progress. (p. 332)
5. **Stop and restart the analysis.** Optionally, you can:
   a. Stop the analysis. (p. 141)
   b. Modify the analysis as needed.
   c. Restart the analysis. (p. 142)
6. **Disconnect the GUI and then reconnect to the running analysis. (p. 139)**
7. **Review analysis output.**
   a. Review System Coupling's output. (p. 330)
   b. Review System Coupling's charting output (p. 197), either in the GUI or in `.csv` output files.
   c. Load System Coupling's analysis results into EnSight (p. 207) for postprocessing.

d. Review the participants' results using the method(s) recommended in the participant product documentation (for example, CFD-Post).

Note:

The steps vary slightly for a coupled analysis based on a setup exported from Workbench.

Expressions in System Coupling

System Coupling supports the use of expressions to specify values for the following data model settings:

- All settings with immutable single-valued real or integer values
- Source-side data transfer values
- Time step size

Expression Definition

You can define an expression using any consistent mathematical combination of the following elements:

- **Values**: These may be single values, field values, or dimensional constants. For more information, see Expression Values (p. 62) and Constants (p. 66).
- **Variables**: These may be participant variables or System Coupling's built-in variables and may have either scalar or vector tensor types. Source and target tensor types must be consistent. For more information, see Variables in Expressions (p. 63).
- **Standard numerical operators and functions**: Mathematical capabilities supported by the Python programming language (essentially, the functionality defined by the C programming standard) can be used in expressions. For more information, see Mathematical Operators, Constants, and Functions (p. 65).
- **Conditional and Boolean statements**: Conditional and Boolean statements can be used to build expressions. For more information, see Conditional and Boolean/Logical Statements (p. 67).
- **Quantity units**: These may be any consistent units supported by System Coupling. Source and target units must be consistent. For more information, see Units in Expressions (p. 68).
- **Python functions**: Python functions can be created, loaded each time the coupled analysis is opened, and used to provide single-valued data in expressions. For more information, see Python Expression Functions (p. 69).
- **Named expressions**: Expressions can be saved and referenced by name in other expressions. For more information, see Named Expressions (p. 73).

Expression Evaluation

Expressions are evaluated as follows:

- Expressions are evaluated during the solution.

- Expression operators are evaluated from left-to-right in order of their precedence (that is, according to standard order-of-operation rules).
- Expressions may evaluate to real or integer single values or field values, which may be either immutable or mutable.
- Values used in mapping and data transfers between participants can be scalars or vectors, consistent with the tensor type of the target variable.
- Expressions that include Python functions are evaluated at least once per solve. For details, see Expression Function Definition, Validation, and Evaluation (p. 69).
- For data transfers, the expression is evaluated on the source side of the interface and then mapped to the target side.

For more information on using expressions in System Coupling, see the following sections:

Expressions Validation

Expression Values

Expression Definitions

Using Expressions to Set Data Model Values

Expressions Validation

Note:

System Coupling's evaluation of expressions is based on the Python 3.10 programming language. For more information, see the resources published by the Python Software Foundation.

When entering an expression, construct it in the form of an operand, followed optionally by operators and additional operands. (For example, if `Temp1` and `Temp2` are existing variables in a coupled analysis, you can define a setting value as **`Temp1 + 3 * Temp2`**).

Ensure that the expression meets the following syntactic requirements, and has:

Syntactically Valid Names

The names used in expressions to reference various coupled analysis objects (which includes variable names, function names, and so on) must adhere to the Python language's naming rules. For example, names:

- must be unique within the analysis
- are case-sensitive
- can be constructed only of letters, digits, and the underscore characters
- cannot have a digit as the first character
- cannot include spaces

- cannot include reserved keywords
- cannot have the same name as a variable (either participant variables or built-in System Coupling variables)

The Expression Name used to define named expressions must also conform to these rules. For more information, see Named Expressions (p. 73).

Supported Variables

Expressions must include only variables that are supported for the data model setting being defined. For example:

- Expressions used to define source-side data transfer variables cannot reference target-side variables.
- In a steady analysis, expressions used to define source-side data transfer values cannot reference time-based variables.
- For expressions used to define a mutable time step size (p. 77), only System Coupling's `Step`, `Time`, and `PreviousTimestepSize` solution variables are supported for use in the expression.

Consistent Variable Locations

Data transfer variables must have a data location definition. When multiple variables are referenced in an expression, all must have `Location` set to the same type (either `Node` or `Element`).

Consistent Variable Tensor Types

Data transfer variables must have a `TensorType` defined (***`Vector`*** or ***`Scalar`***). When multiple variables are referenced in an expression, they are subject to the following constraints:

- Terms that are added to or subtracted from each other must have the same tensor type.
- A vector term may not be multiplied by another vector term.
- A vector and a scalar may be multiplied in either order and results in a vector value in which each component of the original vector has been multiplied by the scalar.
- A vector term may not be divided by another vector term.
- A vector may be divided by a scalar (but not vice versa) and results in a vector value in which each component of the original vector has been divided by the scalar.

Consistent Target Values

Expressions must evaluate to a value that is consistent with the requirements of the data model setting being defined — that is, the evaluated value must match the following characteristics of the target value:

- single-valued or field
- immutable or mutable

- vector or scalar tensor type

For more information, see Expression Values (p. 62).

Dimensional Consistency

Expressions must evaluate to units that are consistent with the units of the data model setting being defined. For example, an expression defining a data transfer source variable must evaluate to units that are dimensionally compatible with those of the target variable.

For more detailed information, see Units in Expressions (p. 68).

Expression Values

Expressions must result in a value that is consistent with the requirements of the data model setting being defined. The evaluated value(s) must match the characteristics of the target value: whether it is a ***single value*** or a ***field***, and whether it is ***immutable*** or ***mutable***.

Single Values

- When an expression evaluates to a single value, it returns a single value at each point of the calculation — that is, there is one value at each coupling step or iteration in the analysis.
- Single values are not dependent on mesh or location.

Fields

- When an expression evaluates to a field, it returns a field of values at a given point in the calculation — that is, at a given point in the calculation, there are values throughout the regions on a coupling interface.
- Fields are dependent on mesh and location, so evaluation of the expression may yield different values for each node or element of the mesh at each coupling step or iteration.

Note:

It is possible to assign a single value to a field. In this case, each location in the field will be set to this value.

Immutable Values

- An expression that does not reference any variables evaluates to an immutable value.
- When an expression evaluates to an immutable value(s), the returned value(s) do not change over the course of the analysis.
- The expression is evaluated once and the value(s) remain the same at each coupling step or iteration.

Mutable Values

- An expression that references one or more variables evaluates to a mutable value.

- When an expression evaluates to a mutable value(s), the returned value(s) vary over the course of the analysis.
- The expression is evaluated multiple times, yielding different values at each coupling step or iteration.

Table 12: Example expression values

Evaluated Value	Setting(s)	Description
Immutable Single Value	**End Time**	An expression defining the **End Time** value is evaluated once. The value does not change during the analysis.
Mutable Single Value	**Time Step Size**	An expression defining a mutable **Time Step Size** value may evaluate to **1 [s]** at a given coupling step and to **2 [s]** at a later step.
Mutable Field	**Value**	An expression defining the source side variable values for a Force transfer will yield different values for each coordinate (for example, `force`) at different coupling steps or iterations. Note also that the components of a vector field may be accessed as scalar fields – `force.x`, `force.y`, and `force.z`.)

Expression Definitions

For details on components that may be used to define an expression, see the following sections:

Variables in Expressions

Mathematical Operators, Constants, and Functions

Conditional and Boolean/Logical Statements

Units in Expressions

Python Expression Functions

Named Expressions

Variables in Expressions

For more information on the use of variables in expressions, see the following sections:

Variable Sources

Data Transfer Variable Tensor Types

Variable Sources

Expressions may reference System Coupling's built-in variables and/or variables provided by participants.

Note:

- Currently, all built-in System Coupling variables have mutable values.
- The `Area` variable is supported only for transfers element data from surfaces. That is, the source participant must have `Region.Topology` set to ***`Surface`*** and `Variable.Location` set to ***`Element`***.
- The `Area` and `Volume` variables are not available for point cloud regions.

System Coupling's Built-in Variables

System Coupling has the following built-in variables, which can be used in expressions:

- **Solution Variables**:
  - `Iteration`
  - `Time`
  - `Timestep`
  - `PreviousTimestepSize`
- **Region Variables**:
  - `Area`
  - `Normal`
  - `Position`
  - `Volume`

For more detailed information on these variables, see Built-In Variables in the *System Coupling Settings and Commands Reference* documentation.

Coupling Participant Variables

Any variable listed by a coupling participant as a System Coupling variable for the region(s) on the coupling interface may be used in expressions.

Data Transfer Variable Tensor Types

System Coupling supports the use of both scalar and vector variables (that is, variables with `TensorType` set to ***`Scalar`*** or ***`Vector`***, respectively) in expressions.

Note:

Vector-valued data transfers are defined by a single vector-valued expression in the data transfer's `Value` setting.

Previously, vector transfers were defined using separate components in the data transfer's `ValueX`, `ValueY`, and `ValueZ` settings. These components are now combined into the single `Value` setting.

For source-side expressions, the tensor type of the target variable determines both how the variable is defined and the tensor type of the source-side expression (that is, the `Value` setting), as follows:

- **If the target variable is scalar:**
  - The source variable is defined as a single value.
  - The source expression must resolve to a scalar value.
- **If the target variable is vector:**
  - The source variable is defined as Cartesian coordinates (`x`, `y`, and `z`). To access the vector components of a variable, enter the variable name followed by a period ('`.`') and the vector component name. For example:

    ```
    Force.x
    Force.y
    Force.z
    ```

  - The source expression must resolve to a vector value.

Mathematical Operators, Constants, and Functions

The following sections detail System Coupling's support for mathematical operators, constants, and functions:

Operators

Description	Setting	X Units	Y Units
Negation	`-x[a]`	any	

Description	Setting	X Units	Y Units
Addition	`x[a] + y[b]`	any	`[a]`
Subtraction	`x[a] - y[b]`	any	`[a]`
Multiplication	`x[a] * y[b]`	any	any
Division	`x[a] / y[b]`	any	any
Exponent	`x[a] ** y`	any	dimensionless

Boolean/Logical Operators

Description	Syntax	X Units	Y Units
NOT	`not x`		
Less Than or Equal to	`x[a] <= y[b]`	any	`[a]`
Less Than	`x[a] < y[b]`	any	`[a]`
Greater than or equal to	`x[a] >= y[b]`	any	`[a]`
Greater than	`x[a] > y[b]`	any	`[a]`
Not equal to	`x[a] != y[b]`	any	`[a]`
Equal to	`x[a] == y[b]`	any	`[a]`
AND	`x and y`		
OR	`x or y`		

Constants

Description	Symbol	Value
pi	`pi`	3.1415927
Base of natural logarithm	`e`	2.7182818

Functions

Description	Setting	X Units	Y Units
Modulus	`x % y`	dimensionless	dimensionless
Floor Divide	`x[a] // y`	any	`[a]`
Integer	`int(x)`	dimensionless	
Nearest Integer	`nint(x)`	dimensionless	
Ceiling	`ceil(x)`	dimensionless	
Absolute Value	`fabs(x[a])`	any	
Absolute Value	`abs(x[a])`	any	
Floor	`floor(x[a])`	dimensionless	
e^x	`exp(x)`	dimensionless	
Natural logarithm	`log(x)`	dimensionless	
Base 10 logarithm	`log10(x)`	dimensionless	
Power	`pow(x[a], y)`	any	

Description	Setting	X Units	Y Units
Square root	`sqrt(x[a])`	any	
Arc cosine[1] (p. 67)	`acos(x)`	dimensionless	
Arc sine[1] (p. 67)	`asin(x)`	dimensionless	
Arc tangent[1] (p. 67)	`atan(x)`	dimensionless	
Arc tangent with quadrant[1] (p. 67)	`atan2(x[a], y[b])3`	any	`[a]`
Cosine	`cos(x)`	angle	
Sine	`sin(x[a])`	angle	
Tangent	`tan(x[a])`	angle	
Hyperbolic cosine	`cosh(x)`	dimensionless	
Hyperbolic sine	`sinh(x)`	dimensionless	
Hyperbolic tangent	`tanh(x)`	dimensionless	
Minimum	`min(x[a], y[a])`	any	`[a]`
Maximum	`max(x[a], y[a])`	any	`[a]`
Conditional[2] (p. 67)	`x[a] if <log_expr> else y[b]`	any	`[a]`

1: The results of inverse trig functions are unitless if the arguments are unitless. You must perform a mathematical operation on the result to add units if you need them. For example, multiply the result of the function by 1 [radian].

2: The Conditional function is supported for the following structural and thermal loads: Pressure, Moment, Force (when defined by force per-unit-area and directional components), Heat Flux, Temperature, and Convection (on ambient temperature).

Conditional and Boolean/Logical Statements

System Coupling supports the use of conditional, Boolean, and Logical statements in expressions. These statements can be used to build expressions, which in turn can be used as operands in other expressions.

Conditional statements

Conditional statements use the `if-else` construct to test a condition and optionally provide actions to be taken if the condition is met and/or is not met.

Boolean/Logical statements

Boolean/Logical statements use Boolean and numeric Logical operators to set variable values.

Example 1: Expressions using conditional and Boolean/Logical statements

```
500 [N] if diameter >= 15 [mm] else 250 [N]
500 [N] if diameter >= 15 [mm] else 250 [N]
```

Units in Expressions

Dimensional quantities are defined in units that can be a combination of one or more base units. Expressions must have compatible units, as follows:

Addition and Subtraction operators

The addition and subtraction (**+** and **-**) operators require that the two operands have compatible units. For example, you cannot add or subtract an **Area** parameter and a **Length** value. Both units must be **`Length`** or both units must be **`Area`**.

However, you can add or subtract two dissimilar units that are of the same dimensionality. For example, **`1 [m] + 1 [ft]`** is a valid expression because both units are of the quantity type **`Length`**.

Multiplication and Division operators

The multiplication and division (***** and **/**) operators do not require that operands have the compatible units. They allow the following permutations:

- One operand can be a quantity with a unit and the other operand can be a dimensionless factor.
- Both operands can be quantities with units, where the result is a different quantity type. For example, **Length/Time** results in units of equivalent dimension to a **`Velocity`** unit.

Trigonometric functions

The resulting units for arguments to a trigonometric function may be dimensionless or resolve to an angle, which allows for the following permutations:

- If the argument is a number or dimensionless, then the value is interpreted as radians. For example:

  ```
  1 [] = 1 [rad]
  ```

- If the argument is a quantity, then the units must resolve to type **`Angle`**. For example:

  ```
  1 [rev] = 360 [deg] = pi * 2[rad]
  ```

Whole expressions

You can apply a unit to a whole expression by multiplying it by a dimensional constant. For example:

```
sin(0.1) * 1 [N]
```

Note:

The notation **`sin(0.1) [N]`** is invalid because the syntax is ambiguous in Python.

For general information on supported units, see Units in System Coupling (p. 39).

Python Expression Functions

You can reference external Python functions in expression definitions. To make a function available for use in expressions, you must first define it in a Python module and then add it to the coupled analysis as an expression function object.

When you add the first expression function object, the `Library` singleton is added to the data model (if it does not already exist), with the `ExpressionFunction` object container and the new expression function object defined beneath it.

Note:

The singleton is shown in the GUI as the **Library | Expression Function** branch.

Once the object is added to the data model, the function is loaded each time the analysis is opened and you can reference the function by name in expressions. Once in use, an expression function cannot be deleted.

For more information, see:

Expression Function Definition, Validation, and Evaluation

Defining an Expression Function

Adding an Expression Function

Reloading Expression Function Modules

Expression Function Definition, Validation, and Evaluation

Function Definition

- Functions are defined in one or more Python module files.
- Python modules must be placed in a user-created **Modules** subdirectory in System Coupling's working directory.
- Defined functions added to the coupled analysis are available for use in expressions.
- Function names referenced in expressions may be different than the Python names of the functions.
- Function arguments referenced in an expression are dimensionless and return dimensionless values.

Function Validation

- Expression modules and functions must follow Python's syntax and naming rules.
- Expression function parameters must be unique within the analysis.
- The number of arguments passed to a function in an expression must match the number of parameters defined for the function.
- Functions are expected always to return the same value for given argument values.

- A function that is called with no arguments, or with arguments that do not depend on any variables, is assumed to return an immutable value and may be referenced in an expression that resolves to an immutable value.
- Inconsistencies between modules and expression definitions may cause validation errors. For more information, see Reloading Expression Function Modules (p. 72).

Function Evaluation

- If a function does not reference any variables in its arguments (or has no arguments), then the expression is treated as if it is returning an immutable value. As such, it can be referenced in expressions resolving to an immutable value and will be evaluated at least once per solve.
- If a function references a variable in its arguments, it will be called when the variable value updates (per iteration for iteration-based analyses or per step for step-based analyses).

Defining an Expression Function

Expression functions are defined in a Python module. To define an expression function, perform the following steps:

1. Create the module.
   a. In System Coupling's working directory, create a **Modules** subdirectory.

      This is where System Coupling looks for Python expression modules and functions.

   b. Inside the subdirectory, create a Python module (for example, a file called `myModule.py`).

2. Define the function.
   a. In the file, define the function, providing arguments for the function name and definition. The example below defines a function called `f`.

```
def f(step):
    return 2*step
```

   b. Save the file.

The function may now be added to the data model.

Adding an Expression Function

To make a function available for use in expressions, you must add an expression function object to the data model. This can be done using either the GUI or the CLI.

Adding an Expression Function in the GUI

1. Perform one of the following actions:

   - From the **Menu Bar**, select **Setup** → **Add Expression Function**.
   - In the **Outline** tree, right-click the **Setup** branch and select **Add Expression Function**.

   The **Expression Function** branch is added to the **Library** branch, with the new expression function defined underneath it.

   Note:

   Once the **Expression Function** branch exists, you may add new expression functions by right-clicking the branch and selecting **Add**.

2. Click the expression function.

   Corresponding settings are shown in the **Properties** pane below.

3. Set values for the function to be added, as follows:

   a. **Module**: Select name of the module containing the function to be added. The drop-down is populated with the names of Python modules in System Coupling's **Modules** directory.

   b. **Function**: Select the name of the Python function to be added. The drop-down is populated with the functions defined in the module specified by the **Module** setting.

   c. **Function Name**: (Optional) Type in the name to be used when adding the function to an expression. If a name is not selected, then this setting defaults to the Python function name selected for the **Function** setting.

Adding an Expression Function in the CLI

Run the `AddExpressionFunction()` command, using the `Module` argument to specify the module containing the function to be added and the `Function` argument to specify the Python name of the function to be added.

Optionally, you may use the `FunctionName` argument to specify the name to be used when adding the function to an expression. If a name is not specified, then this setting defaults to the Python function name specified by the `Function` argument.

```
>>> AddExpressionFunction(
 Module = 'myModule',
 Function = 'f',
 FunctionName = 'stepFunction')
```

When the command is run, the expression function object is added under the `Library` singleton.

Once the expression function object has been added to the data model, you can reference the function by name in expressions. Once in use, an expression function cannot be deleted.

For more detailed information, see `ExpressionFunction` in the *System Coupling Settings and Commands Reference* documentation.

Reloading Expression Function Modules

By default, expression function modules are loaded and applied to a coupled analysis only once per System Coupling session. Because of this, changes made to modules during a session are not applied to the current session automatically and are instead applied the next time the coupled analysis is opened. However, changes to modules made during a session are saved to any snapshots that are created.

System Coupling generates a warning when it detects the following issues:

- **There is a discrepancy between a module and analysis expression definitions.**

  The module file has been modified since it was last loaded and may no longer be consistent with the expression definitions in the coupled analysis. Depending on the nature of the modifications, this could result in unexpected errors or differences in results.

- **Expression function modules have failed to load.**

  One or more modules in the **Modules** directory have failed to load or reload to the coupled analysis. This check is performed on all modules, regardless of whether their functions have been added to the analysis.

To synchronize the modules and expression definitions, reload the modified modules from either the GUI or the CLI.

Reloading Modules in the GUI

When module issues are detected, a warning is written to the **Messages** tab.

In the **Outline** tree, right-click the **Library | Expression Function** branch and select **Reload Expression Function Modules**.

Reloading Modules in the CLI

When module issues are detected, a warning is written to the CLI.

Run the `ReloadExpressionFunctionModules()` command.

The modules are reloaded to the coupled analysis.

Note:

When you delete a function from a module and then reload the module, the deleted function is not removed from the coupled analysis. If the deleted function is refer-

enced by an expression function object and the object is used in an expression, no error will be shown until the analysis is saved and then reopened in a new System Coupling session.

Named Expressions

For ease of reference, you can add named expressions that can be incorporated into other expression definitions. To make an expression available for use in other expressions, you must add a named expression object to the data model.

When you add the first named expression object, the `Library` singleton is added to the data model (if it does not already exist), with the `Expression` object defined beneath it.

Note:

The object container is shown in the GUI as the **Library | Named Expression** branch.

Named expressions follow the naming and definition rules outlined in Expressions Validation (p. 60).

For more information, see:

Adding a Named Expression

Using a Named Expression

For more detailed information, see `Expression` in the *System Coupling Settings and Commands Reference* documentation.

Adding a Named Expression

Named expressions may be added either in the GUI or the CLI, as follows:

Adding a Named Expression in the GUI

1. Perform one of the following actions:

   - From the **Menu Bar**, select **Setup** → **Add Named Expression**.
   - In the **Outline** tree, right-click the **Setup** branch and select **Create Named Expression**.

   The **Named Expression** branch is added to the **Library** branch, with the new expression defined underneath it.

   Note:

   Once the **Named Expression** branch exists, you may add named expressions by right-clicking it and selecting **Add**.

2. Click the named expression.

   Settings are shown below in the **Properties** pane.

3. Enter the expression as the value of the **Expression String** setting.

Adding a Named Expression in the CLI

Run the `AddNamedExpression()` command, using the `ExpressionName` argument to specify a display name for the expression and the `ExpressionString` argument to specify its definition.

```
>>> AddNamedExpression(
        ExpressionName = 'HeatRateExpr'
        Expression String = 'heatrate/5')
```

When the command is run, the expression object is added under the `Library` singleton.

Using a Named Expression

Once the named expression object has been added to the data model, you can reference the expression by name in other expressions by right-clicking the setting's value field and selecting **Insert named expression**. Once in use, a named expression cannot be deleted.

Using Expressions to Set Data Model Values

For instructions on creating expressions and using them to set data model values, see the following sections:

Creating an Expression for an Immutable Single Value

Identifying Valid Expression Variables for Mutable Values

Creating an Expression for a Mutable Time Step Size

Creating an Expression for a Data Transfer Source Variable

Creating an Expression for an Immutable Single Value

You can use either the GUI or the CLI to create an expression for any data model setting with an immutable single real or integer value.

Creating an Expression for an Immutable Single Value in the GUI

1. Under **Setup**, locate the setting to be defined.
2. In the **Properties** pane, enter the expression as the value for the setting.

> **Note:**
>
> If one or more named expressions are defined, you can right-click the setting's value field and select **Insert named expression**.

Creating an Expression for an Immutable Single Value in the CLI

In the CLI, enter the path to the setting, then enter an equal sign (=) and type in the expression string inside single quotes.

Identifying Valid Expression Variables for Mutable Values

System Coupling's variables can be used to define expressions for settings with mutable values.

Note:

Currently, the only supported settings with mutable values are:

- Data transfer source variables
- `TimeStepSize` (when `AnalysisType` is ***`Transient`*** and the `DurationOption` is set to ***`EndTime`***)

The units resulting from a source expression must be consistent with the target variable's units. Thus, to add a variable to an expression defining the source variable for a data transfer, you must identify both the name and units of source-side variables that are valid for use in expressions. You can obtain this information either from the GUI or by running a command.

Both the GUI and command-line methods provide the following information:

- A way to identify the target units
- A list of all variables that are available and valid for use in a source expression on the selected object

  In the list, variables with the same quantity type and/or dimensionality as the target of the expression are listed first. Any of the variables returned may be included in an expression, provided that other validation criteria do not prevent their use.

Note:

The source and target units are not required to match, so long as appropriately dimensioned adjustment factors are used in the expression. The source expression needs only to resolve to units that are dimensionally consistent with the target units, as described in Units in Expressions (p. 68).

Identifying Expression Variables for Mutable Values in the GUI

Right-click inside the expression field, select **Insert variable**, and select a variable from the drop-down menu. The menu is populated with the variables available for the setting.

When a variable is highlighted in the drop-down, its units are shown in a tooltip.

Identifying Expression Variables for Mutable Values in the CLI

Run the `PrintExpressionVariables()` command as illustrated in the examples below to obtain a list of available variables in table format.

Example 2: Use with ObjectPath to get source-side variables for a data transfer

```
>>> PrintExpressionVariables(ObjectPath = '/SystemCoupling/CouplingInterface:Interface-1/
DataTransfer:Transfer-1')

+-------------------------------------------------------------------------------+
|                           Available Variables                                 |
+-------------------------------------------------------------------------------+
| Variable                            |   Tensor Type        Dimension        |
+-------------------------------------------------------------------------------+
| Target                              |                                       |
|   Loss                              |     (Scalar)       [kg m^2 s^-3]      |
| Source                              |                                       |
|   heatflow                          |     (Scalar)       [kg m^2 s^-3]      |
|   Position                          |     (Vector)            [m]           |
|   Normal                            |     (Vector)            [ ]           |
|   Area                              |     (Scalar)           [m^2]          |
|   Step                              |     (Scalar)            [ ]           |
|   Iteration                         |     (Scalar)            [ ]           |
|   Time                              |     (Scalar)            [s]           |
+-------------------------------------------------------------------------------+
```

Example 3: Use with DatamodelRoot() to get source-side variables for a data transfer

```
>>> DatamodelRoot().CouplingInterface['Interface-1'].DataTransfer['Transfer-1'].
PrintExpressionVariables()

+-------------------------------------------------------------------------------+
|                           Available Variables                                 |
+-------------------------------------------------------------------------------+
| Variable                            |   Tensor Type        Dimension        |
+-------------------------------------------------------------------------------+
| Target                              |                                       |
|   Loss                              |     (Scalar)       [kg m^2 s^-3]      |
| Source                              |                                       |
|   heatflow                          |     (Scalar)       [kg m^2 s^-3]      |
|   Position                          |     (Vector)            [m]           |
|   Normal                            |     (Vector)            [ ]           |
|   Area                              |     (Scalar)           [m^2]          |
|   Step                              |     (Scalar)            [ ]           |
|   Iteration                         |     (Scalar)            [ ]           |
|   Time                              |     (Scalar)            [s]           |
+-------------------------------------------------------------------------------+
```

Creating an Expression for a Mutable Time Step Size

When **Duration Option** is set to **End Time**, you can use an expression to define a mutable time step size for a coupled analysis. This can be done either in the GUI or the CLI.

Note:

- When you create an expression to define a mutable time step size, System Coupling's `Step`, `Time`, and `PreviousTimestepSize` solution variables are the only variables that can be referenced in the expression.
- Regardless of the duration option, you may define an immutable time step size as described in Creating an Expression for an Immutable Single Value (p. 74).

Creating an Expression for a Mutable Time Step Size in the GUI

1. Under **Setup**, click **Solution Control**.
2. In the **Properties** pane below, enter in an expression for the **Time Step Size** setting.

Note:

- To add a System Coupling variable, right-click the field, select **Insert variable**, and select the variable to be used.
- If one or more named expressions are defined, you can right-click the field and select **Insert named expression**.

Creating an Expression for a Mutable Time Step Size in the CLI

Enter the expression as the value of the `TimeStepSize` setting, as shown in the example below.

```
>>> DatamodelRoot().SolutionControl.TimeStepSize = '0.2 if Step < 5 else 0.25'
```

Tip:

To find out what variables may be referenced in the expression, use one of the methods described in Identifying Expression Variables for Mutable Values in the CLI (p. 76).

Creating an Expression for a Data Transfer Source Variable

Expressions can be used to define real values for source-side data transfer variables.

Note:

- Expressions are not supported for data transfer source variables with complex number values (that is, for source variables with `DataType` set to ***`Complex`***).
- Expressions are not supported for data transfers from solvers to FMUs.

When using expressions to define a source-side data transfer variable, keep the following considerations in mind:

- The right-hand side of the expression must resolve to a quantity having the same dimensions as the target variable. If an expression references a source variable with a different `IsExtensive` property than the target variable, then the expression will need to be corrected by area or volume. For example, if transferring a variable with **Force** quantity type (for which `isExtensive` is set to ***`True`***, with dimensions of ***`Newtons`***) to the MAPDL `Force Density` variable (for which `isExtensive` is set to ***`False`***, with dimensions of ***`Force`***/***`Area`***), then the expression must divide by area, as follows:

  ```
  Value = 'Force / Area'
  ```

  Note:

  The exception is when variables are accessed using the **System Coupling** GUI's **Insert Variable** option, in which case the correction is automatically included in the expression.

- In most cases, expressions used in data transfers are used to make minor adjustments to the transferred quantities, such as scaling them up or down based on some physical considerations not included in the model (examples are provided below). It is also possible to have the expression be independent of any variable from the source participant. If doing so, keep the following in mind:

  – In an FSI simulation, you may want to replace the force transferred by the CFD participant with a uniform pressure force. The following example does this for the `Force` target variable, assuming a pressure of ***`2e5 [Pa]`***:

  ```
  Value = '2e5 [Pa] * Area * Normal'
  ```

  – The data transfer expression is evaluated on each mesh location of the interface. If the target variable has the `IsExtensive` property set and you would like to set the integrated quantity over all mesh locations, then you will need to scale the expression to account for it being applied on all mesh locations.

  For instance, suppose the target variable is `Force` (for which `isExtensive` is set to ***`True`***), and you would like to impose a constant total force of ***`1 [N]`*** in the x-direction distributed

by the area associated with each mesh location. If the total area is ***0.01 [m^2]***, then each mesh location will receive a fraction of the total force based on its area, as follows:

```
Value = 'vector(1 [N] * Area / 0.01 [m^2], 0 [N], 0 [N])'
```

The following sections provide instructions on creating an expression to define data transfer source-side variable value(s):

Creating an Expression for a Data Transfer Source Variable in the GUI

Creating an Expression for a Data Transfer Source Variable in the CLI

Creating an Expression for a Data Transfer Source Variable in the GUI

To create an expression for a source-side data transfer variable in the GUI, perform the following steps:

1. In the **Outline** tree under **Coupling Interface**, expand the **Data Transfer** branch.
2. Click the data transfer for which you are defining the source-side variable.

   Corresponding settings are shown below in the **Properties** pane.
3. Set **Option** to **Using Expression**.
4. Set **Target Side** to the side of the interface that is to receive the data transfer, either **One** or **Two**.

   The **Target Variable** setting is automatically populated with the variable associated with the specified interface side.

   Value settings consistent with the **Tensor Type** of the **Using Variable** option's default source variable become available — that is, according to whether the default variable is a scalar or a vector.
5. Set the source variable's value by entering an expression for its **Value** setting, as in the examples shown below:

   - For a variable with a scalar value:

     ```
     Value = '2e5 [Pa] * Area * Normal'
     ```

   - For a variable with a vector value:

     ```
     Value = 'vector(1 [N] * Area / 0.01 [m^2], 0 [N], 0 [N])'
     ```

Tip:

- To add a System Coupling variable, right-click the field, select **Insert variable**, and select the variable to be used.
- If one or more named expressions are defined, you can right-click the field and select **Insert named expression**.

Creating an Expression for a Data Transfer Source Variable in the CLI

You can create source-variable expressions in the CLI either by running a command to add a data transfer or by editing the data model. For details, see:

Creating a Source-Variable Expression by Running a Command

Creating a Source-Variable Expression by Editing the Data Model

---

Tip:

To find out what variables may be referenced in the expression(s), use one of the methods described in Identifying Expression Variables for Mutable Values in the CLI (p. 76).

Creating a Source-Variable Expression by Running a Command

Run the `AddDataTransfer()` command, using the appropriate arguments to create an expression-based data transfer.

Set the source variable's value by entering an expression for its `Value` field, as shown in the example below.

Example 4: Add an expression-based transfer of a vector quantity

```
>>> AddDataTransfer(
 Interface = 'Interface-1',
 TargetSide = 'One',
 TargetVariable = 'INCD',
 Value = 'vector(INCD * Position.x / 1[m], INCD * Position.y / 0.5 [m],
 INCD * Position.z / 0.02 [m])')
```

When the command is run, the `DataTransfer` object is added to the coupling interface.

Creating a Source-Variable Expression by Editing the Data Model

1. For the data transfer for which you are defining the source-side variable, set `Option` to `UsingExpression`.

   ```
   >>> DatamodelRoot().CouplingInterface['Interface-1'].DataTransfer['Temperature-1']
   .Option = 'UsingExpression'
   ```

2. Set `TargetSide` to the side of the interface that is to receive the data transfer, either `One` or `Two`.

   ```
   >>> DatamodelRoot().CouplingInterface['Interface-1'].DataTransfer['Temperature-1']
   .TargetSide = 'Two'
   ```

The `TargetVariable` setting is automatically populated with the variable associated with the specified interface side. Additional settings become available and are determined by the tensor type of the data transfer.

Tip:

To reference a participant variable in the expression, use the value of its `DisplayName` setting when creating the expression.

3. Set the source-side variable's value by entering an expression for its `Value` field, as shown in the examples below.

Example 5: Add an expression-based transfer of a scalar quantity

```
>>> DatamodelRoot().CouplingInterface['Interface-1'].DataTransfer['Temperature-1']
.Value = '100 [K]'
```

Example 6: Add an expression-based transfer of a vector quantity

```
>>> DatamodelRoot().CouplingInterface['Interface-1'].DataTransfer['Force-3']
.Value = 'vector(force.x*1.1, force.y, force.z)'
```

When the edits are applied, the data transfer object is available in the data model under the `DataTransfer` object container.

Participant Solution Sequencing

System Coupling's participant solution sequencing — the order in which participant solutions are executed within an analysis — is based on a combination of participant object names and data transfer ordering requirements. An initial ordering of individual participants is a necessary step for all coupled analyses, both cases where the participants will solve individually and cases where groups of independent participants will be solved simultaneously.

System Coupling performs the following steps to arrange participants into an ordered list:

1. Participants are divided into two sets: those that are sending Incremental Displacement and those that are not, with the participants sending Incremental Displacement coming first.

   This ensures that the final meshes are consistent on source and target participants at the end of each iteration.

2. Within each of the two participant sets, the following order is applied:

a. Participants with a trailing index in their object names (for example, `FLUENT-1`, `MAPDL-2`, `AEDT-3`, `CFX-4`) are first in the list and are sequenced numerically by their indices.

> **Note:**
>
> When participants are added to the coupled analysis, the participant names are automatically appended with a trailing index indicating the order in which participants were added. Other participants appear next and are ordered alphabetically by their object names.

b. Other participants appear next and are ordered alphabetically by their object names.

For coupled analyses where the participants will solve individually (the default behavior), participant data transfers and solutions are executed in the sequence specified by the ordered list.

For coupled analyses where participants are grouped to solve at the same time (the behavior when simultaneous solutions are enabled), additional sequencing steps are performed. For more information, see Simultaneous Execution of Participant Solutions (p. 160).

Working with System Coupling's Data Model

When working in one of System Coupling's user interfaces, you interact with System Coupling by editing its data model in the GUI or the CLI. Once you are familiar with the data model, you can use it to create and/or modify your coupled analysis as described in Creating a Coupled Analysis (p. 120) and Modifying Coupled Analysis Settings (p. 130).

For information on working with the data model, see:

Data Model Structure

System Coupling's data model has a hierarchical structure, much like that of a directory system or file system.

Settings are grouped and contained in paths in the data model. Items defined in the data model tree are of two types: **containers** and **settings**.

Containers

Containers are used to organize the different branches of the data model and can contain other containers and/or settings. There are two types of containers: ***singletons*** and ***object containers***.

- ***Singletons*** are containers that may have multiple instances but can be defined only once within their context. For example:

  - The `AnalysisControl` singleton has only one instance in the data model. Analysis control settings are applied at the analysis level.
  - `ExecutionControl` is a singleton that has multiple instances in the data model, with one defined for each participant. Execution settings are applied at the participant level.

- ***Object Containers*** are containers that may have multiple instances of the same name and that can contain multiple uniquely named objects. For example:
  - There is only one `CouplingParticipant` object container, and it contains multiple named participant objects, such as `MAPDL-1`, `CFX-2`, and so on.
  - There can be multiple instances of the `Region` and `Variable` object containers defined for each coupling participant. They can contain one or more uniquely named region or variable objects, respectively, for the associated participant.

  A coupled analysis includes at least two participant objects and at least one interface object.

Settings

Settings are parameters that have values assigned with an equal sign.

Figure 1: Data model example in System Coupling's GUI

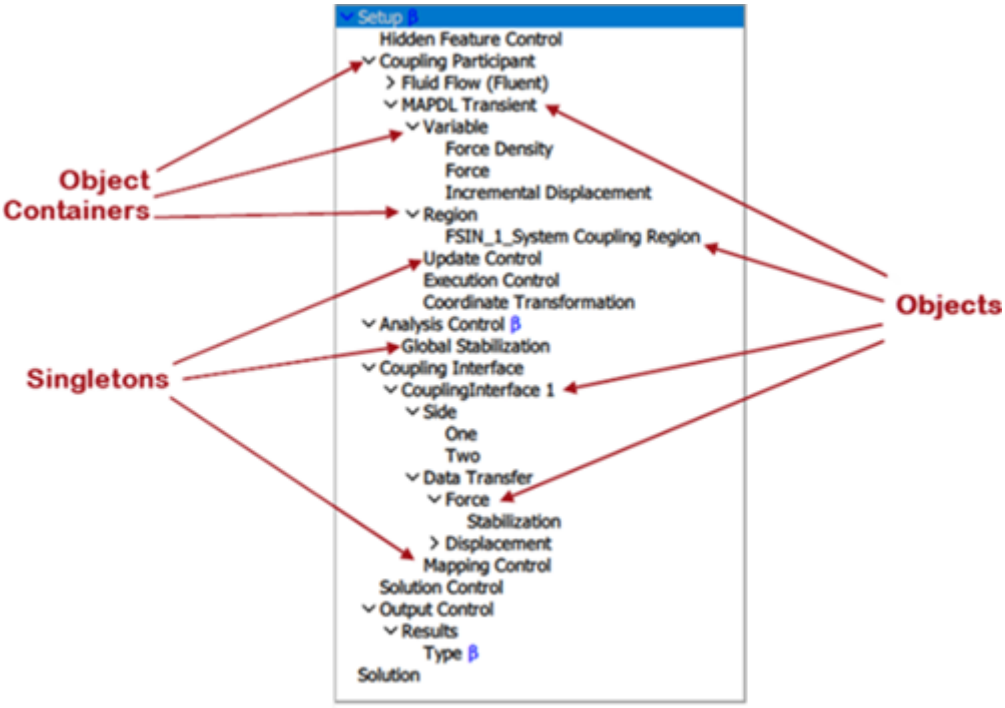


Figure 2: Data model example in System Coupling's CLI

Root-Level Containers

Under the root level of the data model, there are six main containers that control aspects of the coupled analysis:

`Library`

- Contains object containers and global objects that can be referenced elsewhere in the data model.
- Each object container includes settings to define expression, expression function, or transformations.

`CouplingParticipant`

- Contains participant objects.
- Each participant object includes settings for participant type, execution and update controls, regions, and variables.

`AnalysisControl`

- Contains settings that define the coupled analysis and control its behavior.
- Includes settings for analysis type, partitioning for parallel runs, simultaneous participant updates, and stabilization.

`CouplingInterface`

- Contains interface objects.
- Each interface object includes settings for mapping, interface sides, and data transfers.

`SolutionControl`

- Contains settings that define how solutions are executed.
- Includes settings for solution duration, time step size, and number of coupling iterations per step.

`OutputControl`

- Contains settings that determine what coupled analysis output is produced.
- Includes settings for output frequency.

For a comprehensive listing of all data model containers and settings, see the Data Model Settings section of the *System Coupling Settings and Commands Reference* manual.

Viewing Data Model Contents

The following sections provide instructions on different ways to view the contents of the data model:

Viewing a Container's Contents

To view the contents of a container, use the `PrintState()` command on the container. Examples for viewing the contents of singletons (p. 85) and viewing the contents of objects (p. 86) are provided below.

Viewing the Contents of a Singleton

Example 7: View the contents of a singleton

```
>>> DatamodelRoot().SolutionControl.PrintState()

SolutionControl
  DurationOption = EndTime
  EndTime = 0.2 [s]
  TimeStepSize = 0.001 [s]
  MinimumIterations = 1
  MaximumIterations = 10
```

Viewing the Contents of an Object

Example 8: View the contents of a participant's region object

```
>>> DatamodelRoot().CouplingParticipant['MAPDL-2'].Region['FSIN_1'].PrintState()

CouplingParticipant:MAPDL-2
  Region:FSIN_1
    Topology = Surface
    InputVariables = FORC,FDNS
    OutputVariables = INCD
    DisplayName = FSIN_1_System Coupling Region
```

Viewing an Object's Children

To view a list of all of an object's children, run the `GetChildNames()` command on the object.

Example 9: View the children of a coupling participant object

```
>>> DatamodelRoot().CouplingParticipant.GetChildNames()

['MAPDL-1', 'CFX-2']
```

If children exist for the object, then a list of the child names is returned. If children do not exist, then an empty list ('`[]`') is returned.

Viewing the Contents of an Object's Children

To view the contents for all of an object's children, use a Python `for` loop in conjunction with the `GetChildNames()` command.

Note:

When running a command with a `for` loop, in order to complete the loop, you must hit the **Enter** key when presented with "..." in the command console.

Example 10: View the contents of a participant's variable child objects

```
>>> for variable in DatamodelRoot().CouplingParticipant['MAPDL-2'].Variable.GetChildNames():
DatamodelRoot().CouplingParticipant['MAPDL-2'].Variable[variable].PrintState()
...

CouplingParticipant:MAPDL-2
  Variable:FDNS
    QuantityType = Force
    Location = Element
    DisplayName = Force Density
    TensorType = Vector
    IsExtensive = False
```

```
CouplingParticipant:MAPDL-2
  Variable:INCD
    QuantityType = Incremental Displacement
    Location = Node
    DisplayName = Incremental Displacement
    TensorType = Vector
    IsExtensive = False

CouplingParticipant:MAPDL-2
  Variable:FORC
    QuantityType = Force
    Location = Node
    DisplayName = Force
    VariableName = Force
    TensorType = Vector
    IsExtensive = True
```

Using System Coupling's Graphical User Interface (GUI)

System Coupling's graphical user interface (GUI) offers a direct, GUI-based workflow for convenient access to coupling functionality outside the Workbench environment.

For more information, see:

Starting the System Coupling GUI

System Coupling GUI Components

System Coupling GUI Messages and Icons

System Coupling GUI Menus

Starting the System Coupling GUI

You can start the **System Coupling** GUI using either of the following methods:

Starting the GUI from the Start Menu

Starting the GUI from the Command Line

Starting the GUI from the Start Menu

To start the **System Coupling** GUI from the Windows **Start** menu:

1. From the **Start** menu, select **Ansys 2025 R1 > System Coupling 2025 R1**.

   The **Select Folder** dialog opens.

   The **System Coupling** GUI opens. Because you are required to select System Coupling's working directory before starting to populate the data model, the **Select working directory** dialog opens immediately.

2. Navigate to the location that will be the working directory for the coupled analysis, select the directory, and click **Select Folder**.

   Note:

   A valid working directory is required before solving.

The **Select Folder** dialog closes and the **System Coupling** GUI opens in the specified directory.

If the state of a previous coupling run has been saved to the **SyC** directory (that is, there are Settings and Restart files), then System Coupling automatically loads that analysis in its most recent state.

If a solve is already running in this directory, then the GUI connects to it automatically. For details, see Reconnecting the GUI to a Running Coupled Solution (p. 139).

Starting the GUI from the Command Line

To start the **System Coupling** GUI from the command line:

1. Navigate to the directory that is to be System Coupling's working directory.
2. Open the shell for your operating system.
3. Run the System Coupling executable for the appropriate platform, using the `-G` or `--gui`command-line option.

   - **Windows**

     ```
     > "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling" -G
     ```

   - **Linux**

     ```
     $ "$AWP_ROOT251/SystemCoupling/bin/systemcoupling" -G
     ```

The **System Coupling** GUI opens in the current directory and an instance of System Coupling is started. System Coupling runs in this working directory and creates a **SyC** subdirectory for its coupling-related output.

If the state of a previous coupling run has been saved to the **SyC** directory (that is, there are Settings and Restart files), then System Coupling automatically loads that analysis in its most recent state.

If a solve is already running in this directory, then the GUI connects to it automatically. For details, see Reconnecting the GUI to a Running Coupled Solution (p. 139).

Tip:

When starting the **System Coupling** GUI from the command line, you can use the following command-line options to configure the coupling run:

- To specify System Coupling's working directory, you can use the `-w` or `--workDirDialog` option to launch the **Select working directory** dialog.
- To run a script on startup, you can use the `-R` command-line option.
- To specify how long System Coupling waits for participants to connect to the coupled analysis, use the `-C` or `--connectiontimeout` argument, as described in Setting a Participant Connection Timeout Interval (p. 158).
- To control System Coupling's behavior for parallel runs, use the parallel startup options described in Parallel Options for System Coupling (p. 184).

For a comprehensive listing of all available command-line options, see Command-Line Options in the *System Coupling Settings and Commands Reference*.

System Coupling GUI Components

The **System Coupling** GUI includes the following components:

Menu Bar

Toolbar

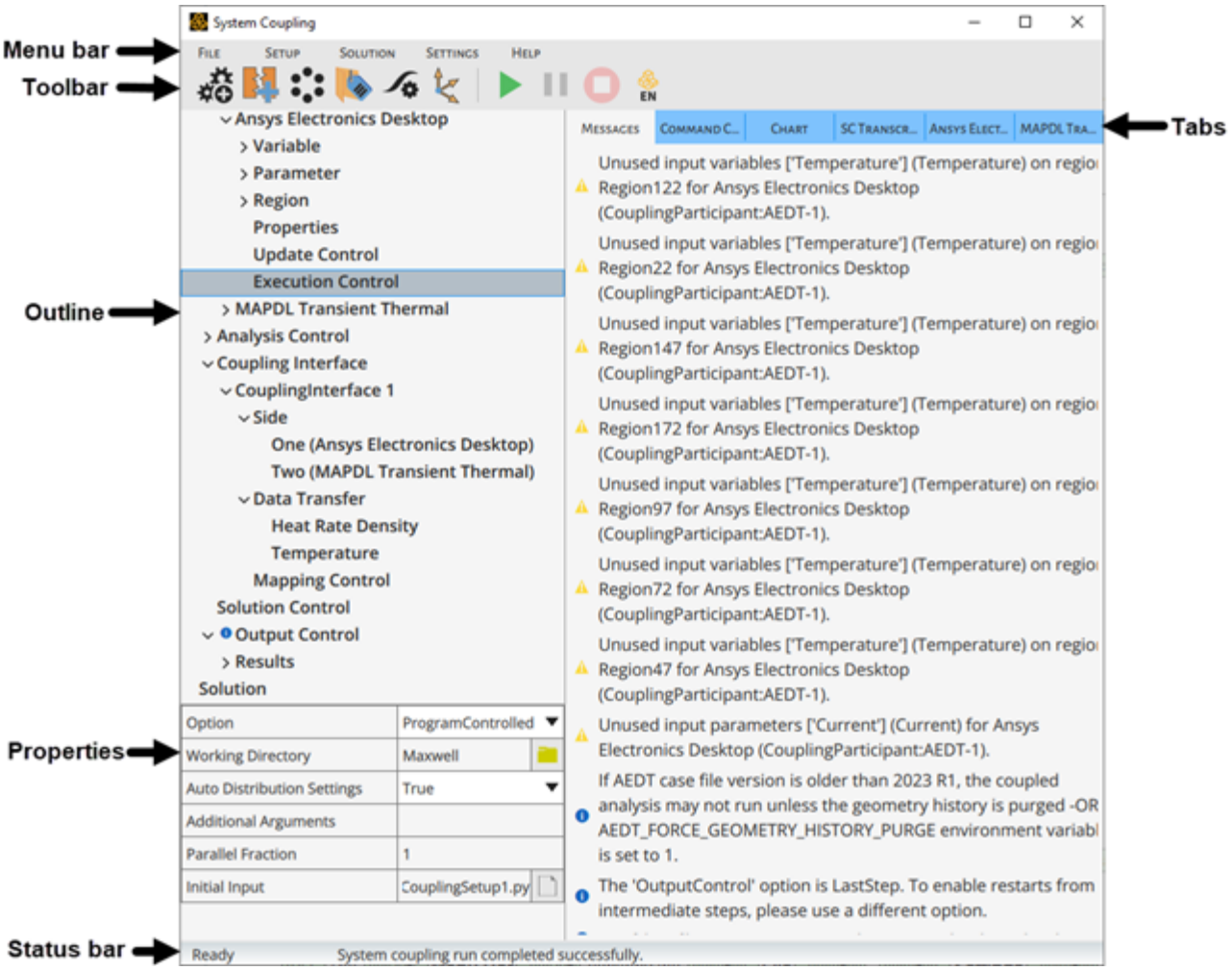
Outline

Properties

Tabs

Status Area

Figure 3: The System Coupling GUI



Menu Bar

The **Menu Bar** contains the **File**, **Setup**, **Solution**, **Settings**, and **Help** menus.

For details on available options, see Menu-Bar Menus (p. 103).

Toolbar

The **Toolbar** includes buttons for the following operations. These buttons are equivalent to the corresponding menu bar (p. 103) and context menu options (p. 109).

Buttons are displayed and enabled according to the type and state of the analysis. Buttons with an arrow to the right of the icon contain a sub-menu. Click the arrow to access these options.

Button	Function	Sub-menu
	Add Participant	N/A

Button	Function	Sub-menu
	Add Interface	N/A
	Add Data Transfer	• Add FSI Data Transfers • Add Thermal Data Transfers • Add Aerodamping Data Transfers • Add Ordered Data Transfers
	Add Instancing	N/A
	Add Named Expression	N/A
	Add Expression Function	N/A
	Add Reference Frame	• Add Reference Frame • Add Rotation Transformation • Add Translation Transformation
	Parallel Solve Setup	N/A
	Solve	N/A
	Preview Mapping	N/A
	Map	N/A
	Interrupt	N/A
	Abort	N/A
	Open Results in EnSight	N/A

Outline

The **Outline** pane provides a tree view for object-focused management of the System Coupling workflow. This pane mirrors both the structure of the data model and the general sequence of steps in a coupled simulation. It is divided into two main branches, **Setup** and **Solution**. Each of these has child branches corresponding to the hierarchical organization of the data model's containers and objects.

Information shown in the tree corresponds to entries shown on the **Messages** tab. When a message applies to a branch of the tree, an icon is shown to the left of the affected branch. The icon indicates the type of information being reported and shows additional details when left-clicked. For more information, see System Coupling GUI Messages and Icons (p. 100).

Setup

The **Setup** branch is used to define data model content, including the addition of new objects as part of the setup process.

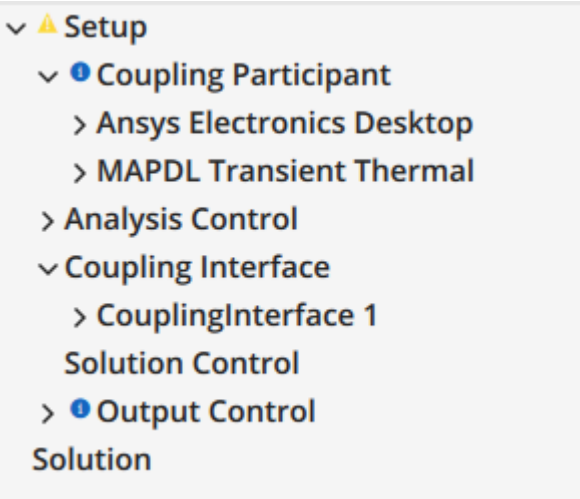
Setup issues that cause data model invalidation are noted on the **Messages** (p. 94) tab.

Solution

The **Solution** branch is used to control the execution of the coupled analysis — to start the solution, to interrupt or abort it once started, and to open EnSight results once completed.

Parent branches can be expanded to show child branches, which may correspond to singleton containers, object containers, or objects. Parent branches with child containers can be expanded/collapsed to show their child containers. Settings under a parent branch are not represented in the tree but are shown in the **Properties** (p. 92) pane when the parent branch is selected.

Figure 4: Outline



Properties

The **Properties** pane shows the settings associated with the object/branch selected above in the **Outline**, with the setting's name on the left and its value on the right. You can edit setting values by selecting or multi-selecting options from a menu and by typing new values into a text field.

For settings with text fields, invalid entries are indicated by red highlighting of the field and/or by messages on the **Messages** (p. 94) tab.

For settings requiring menu selection:

- You can access a drop-down menu using the green down-arrow icon (▼) to the right of the field.
- For the **Region List** setting, if the list of available regions is long enough to require scrolling, you can access a **Select Region** control using the green ellipsis icon (•••) to the right of the field.

Figure 5: Properties for a data transfer

Internal Name	DataTransfer 2
Suppress	False ▼
Target Side	One (Ansys Electronics D... ▼
Option	UsingVariable ▼
Source Variable	Temperature ▼
Target Variable	Temperature ▼
Ramping Option	None ▼
Relaxation Factor	1
Convergence Target	0.01
Mapping Type	ProfilePreserving

Tabs

When a tab is selected, its contents are shown in the area below the tab. The **System Coupling** GUI has the following tabs:

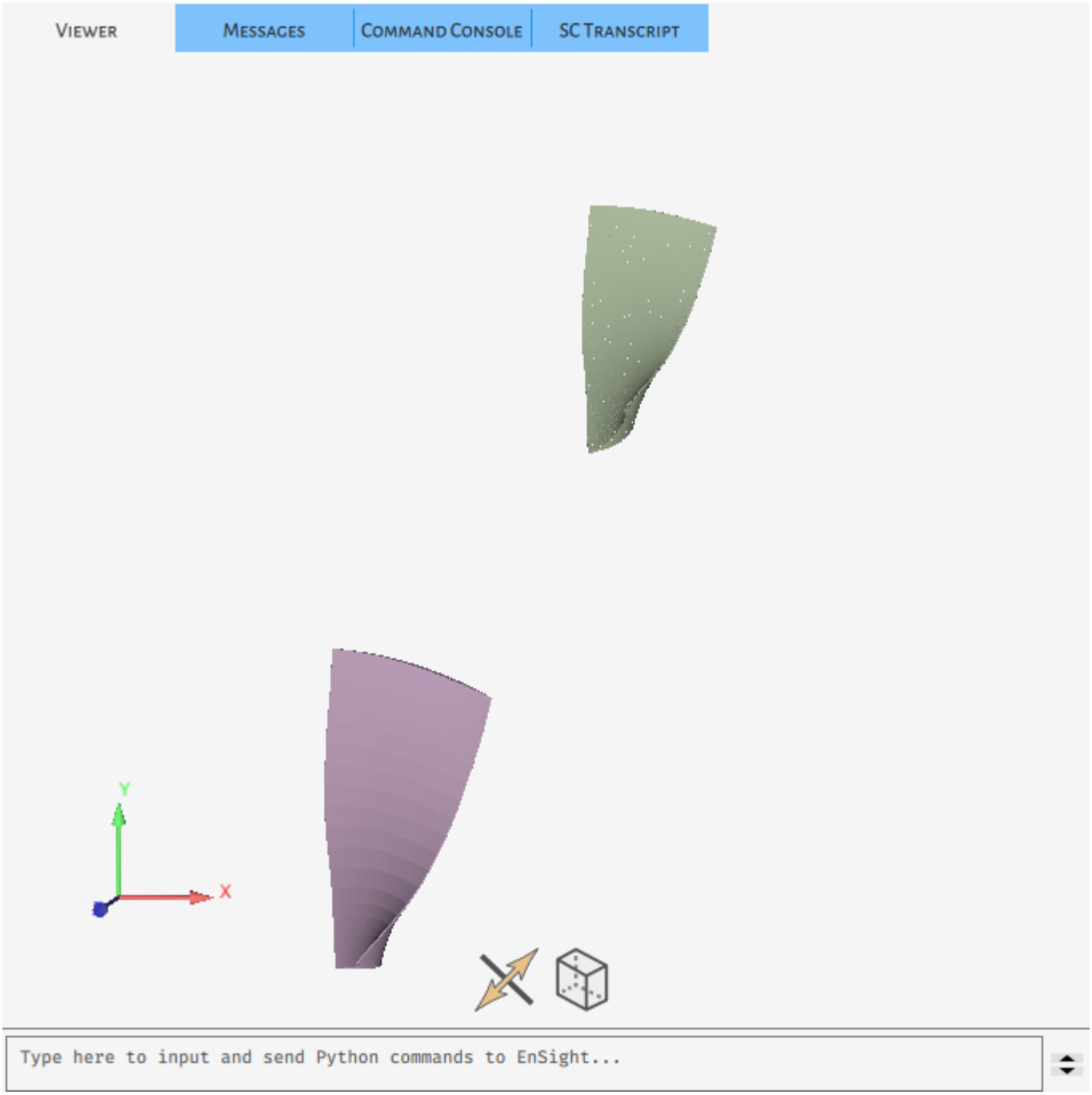
Viewer

The **Viewer** tab shows live visualization (p. 217) results. It is displayed when a live visualization object is created and the **Show Viewer** setting is set to **True**.

You can rotate, zoom in, and zoom out on the model using your mouse. You can also use the following controls:

Control	Icon	Description
Toggle Mesh		Turns the mesh lines on and off.
Fit to Screen		Fits the mesh to the **Viewer** tab area.

Figure 6: Viewer tab



Messages

Displayed by default. The **Messages** tab shows a list of state and/or informational messages for the coupled analysis. Information provided on the **Messages** tab is also reflected in the affected branch(es) of the **Outline** tree.

Each message includes an icon that indicates both the type of information being reported and its level of importance. By left-clicking the icon, you can navigate to the corresponding branch of the tree. For more information, see System Coupling GUI Messages and Icons (p. 100).

Command Console

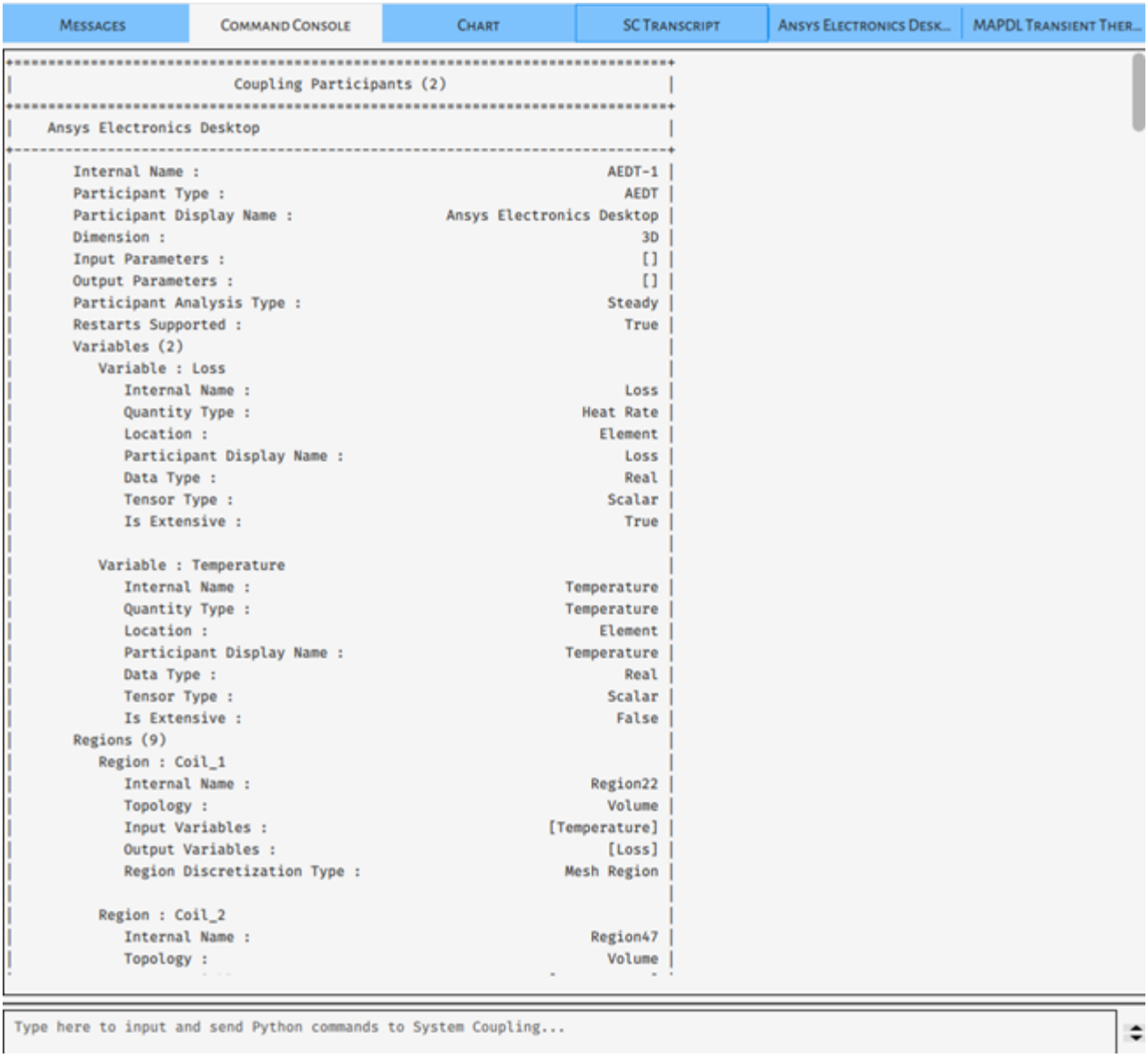
The **Command Console** tab is a Python console that enables you to interact with the data model via command line. The console is divided into two sections:

- The body of the tab shows standard System Coupling output resulting from actions taken in the GUI. Output written here is not retained — if a solved case is saved and then restarted, then the output for the original solve will not be available.
- The bottom of the tab has a **Command Input** area featuring a command line for entering and running Python commands.

Tip:

- System Coupling uses Python 3.10. To use the GUI's **Command Console** tab effectively, you should have some familiarity with working from your operating system command prompt and a basic knowledge of the Python language. For more information, see the resources published by the Python Software Foundation.
- All sub-processes that are manually started from the **Command Console** tab will make use of the system environment that is setup for that console. Review and edit the environment passed to such sub-process if you encounter any problems with their execution.

Figure 7: Command Console tab



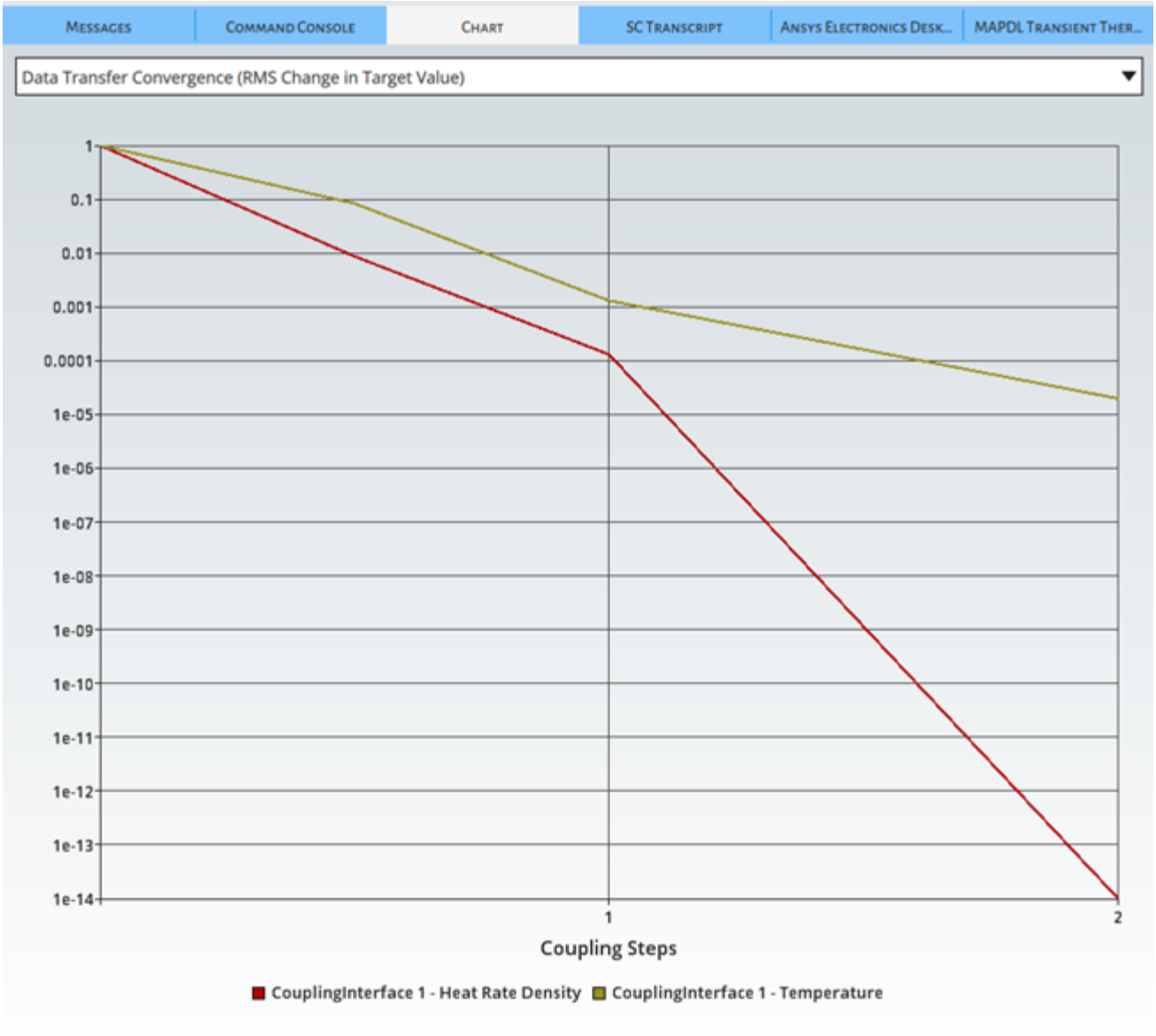
Chart

The **Chart** tab shows convergence data for the analysis. It is shown by default when a solution is started and relevant data becomes available, so can be used to monitor convergence both during a run and after a coupling run.

Data is plotted on an auto-scaled XY line graph. The tab includes a chart selection menu, a legend of the chart data, and a menu for performing different chart operations.

For more information, see Reviewing Convergence Diagnostics Charting Output (p. 197).

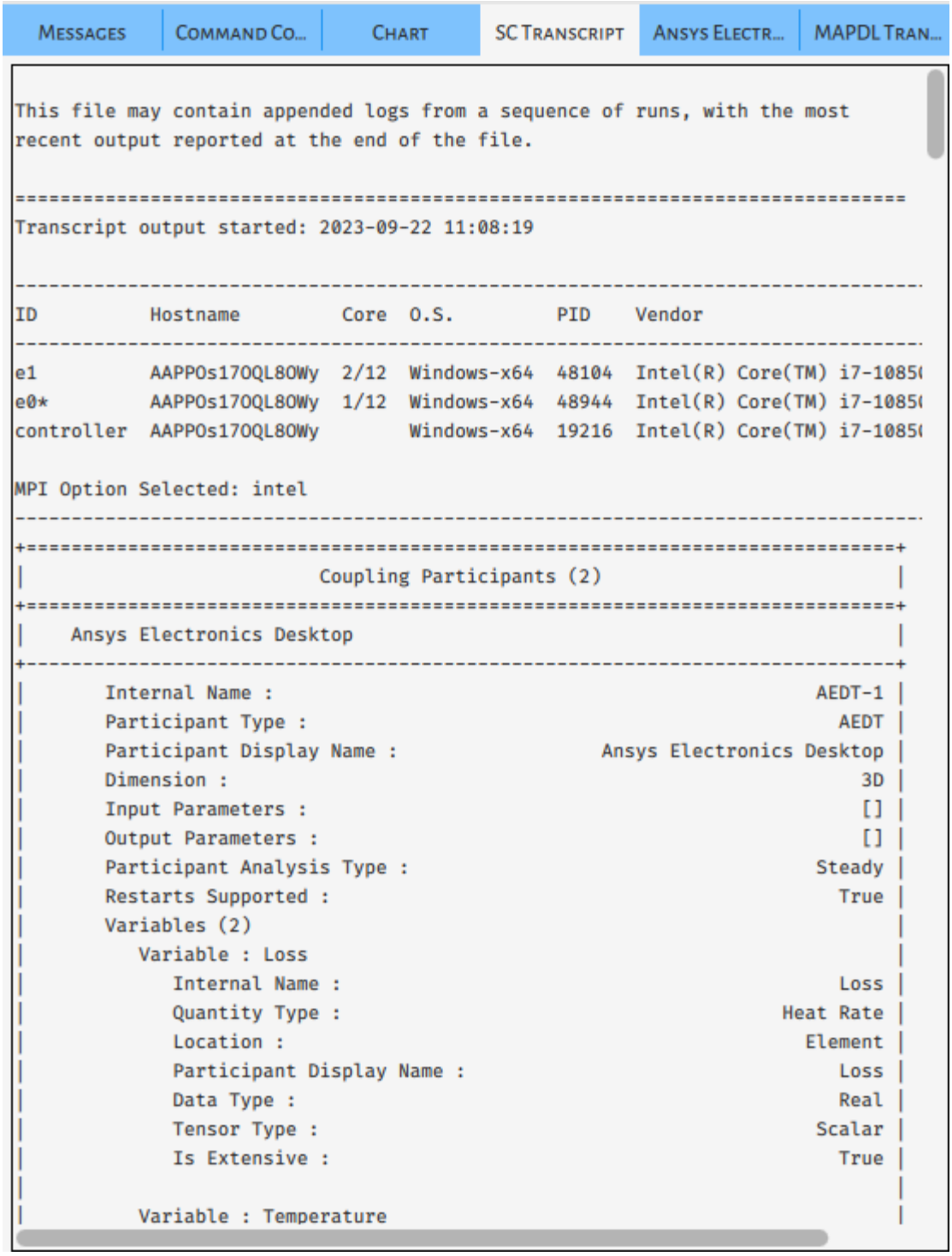
Figure 8: Chart tab



SC Transcript

The **SC Transcript** tab shows System Coupling's transcript output for the coupled analysis. Output written to this tab is retained — if a solved case is saved and then restarted, then the output for the original solve will still be available.

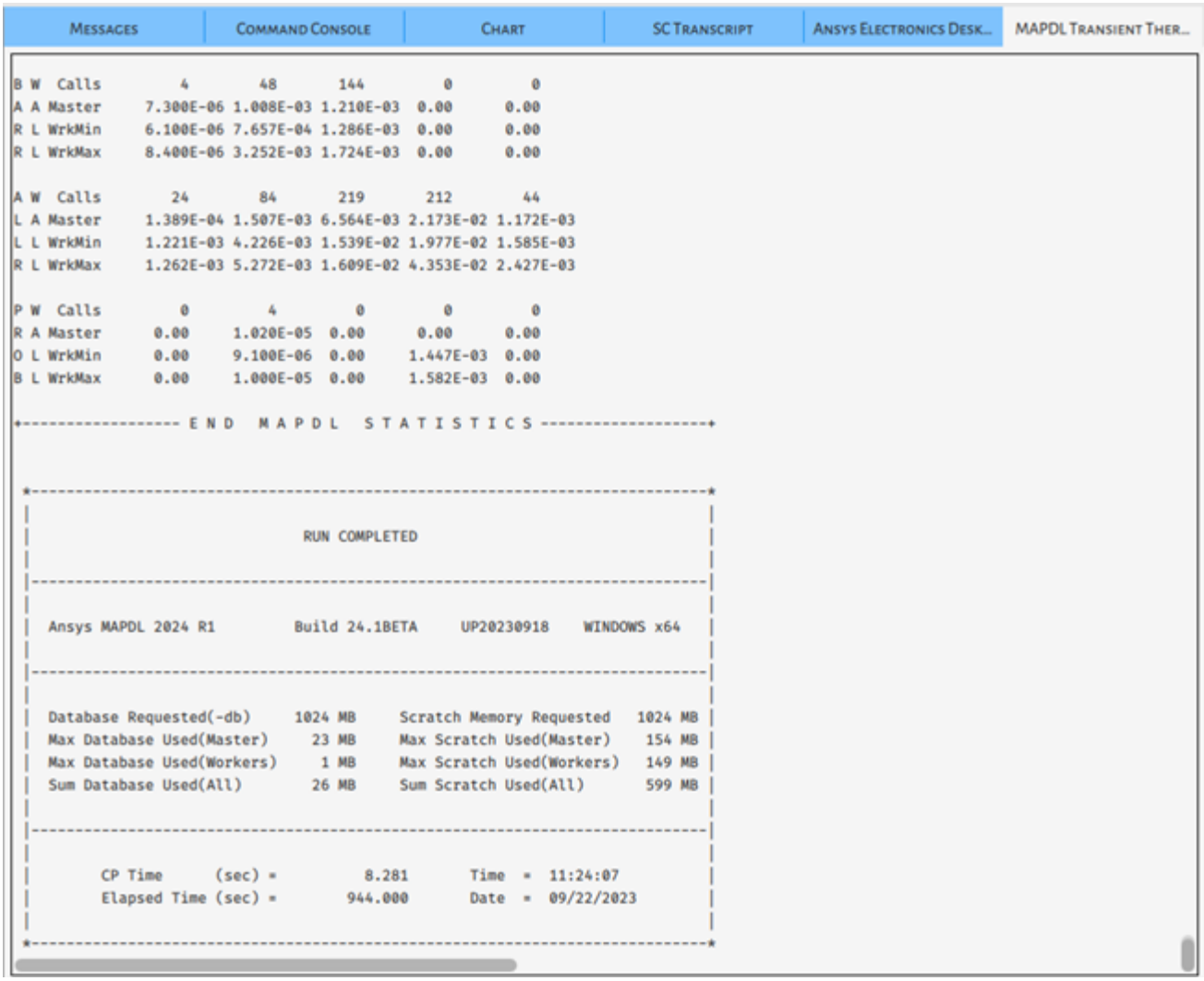
Figure 9: SC Transcript tab



Participant Transcript

The **Participant Transcript** tabs show the transcript output for each coupling participant in the coupled analysis, with one tab per participant.

Figure 10: Participant transcript tab



Status Area

Indicates the status of the coupled analysis.

- **On the left side**, a status message indicates the status of the System Coupling engine. Possible messages are shown in the table below:

  Table 13: Status messages

Message	Description
Ready	System Coupling is waiting for input.
Busy	System Coupling is busy (but not actively solving).
Solving	System Coupling is solving.

- **In the center**, additional messages provide further details about the reported status. For instance, it indicates when System Coupling is waiting for participants to connect, when participants have

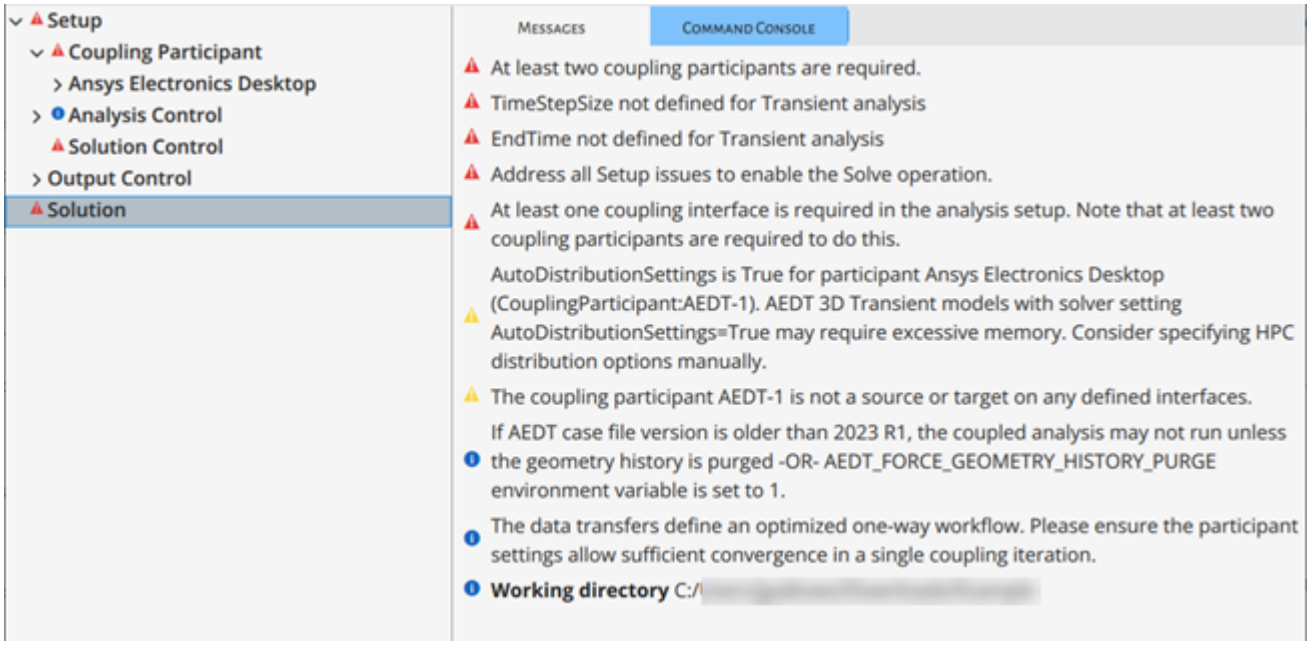
been connected, the percentage of the analysis that's been completed, when the analysis has been stopped and for what reason, and so on.

- **On the right side**, a progress bar shows the progress of the running solution.

System Coupling GUI Messages and Icons

The **System Coupling** GUI's **Outline** tree and **Messages** tab uses a system of messages and corresponding icons to convey relevant information about the state of a coupled analysis, and when necessary, provide guidance on how to address any issues that may exist.

Figure 11: System Coupling GUI's Messages tab



State and hidden-feature messages relevant to the coupled analysis may be displayed. Information shown on the **Messages** tab is reflected in the corresponding branches of the **Outline** tree, as described below:

Messages Tab

The **Messages** tab shows two types of entries: ***state messages*** and ***hidden feature messages***. An icon is shown to the left of each message.

State Messages

- A message is shown for each setting that may require attention.
- When multiple messages of one type are applicable to a setting, the icon reflecting the highest level of importance is shown.
- State messages are sorted first according to the order the corresponding branches are shown in the **Outline** tree, and then according to the level of action needed/possible, with items that require action listed first.

- When you left-click a message's icon or text, the corresponding setting is selected in the **Properties** pane, allowing you to navigate directly to the affected part of the data model.

Hidden Feature Messages

- A message indicates when hidden settings (either beta or alpha) are available for the analysis.
- A message is shown for each hidden setting with a changed default value.
- Messages are sorted with beta-setting messages listed before alpha-setting messages.

Outline Tree

In the **Outline** tree, icons indicate when a branch contains one or more settings with an associated message. Two icons may be shown for each branch: one state icon and one hidden-feature icon (shown to the left and right of the branch label, respectively).

When multiple messages of one type are applicable to settings within the branch, the icon reflecting the highest level of importance is shown.

Properties Pane

When a message on the **Messages** tab is applicable to a setting, left-clicking the message text or icon selects the branch containing the setting in the **Properties** pane.

Two icons may be shown for each setting: one state icon and one hidden-feature icon (shown to the left and right of the setting label, respectively).

When multiple state issues exist for a given setting, then the icon reflecting the highest level of importance is shown.

To learn more about the state of a setting, left-click its state icon. A message provides additional details on the nature of the issue being reported, and, when applicable, offers guidance on how to proceed when corrective action is required.

The table below classifies System Coupling messages and their corresponding icons by type, importance, and function.

Table 14: System Coupling message and icon classifications

Type	Icon	Importance	Description
State		**Action Required**	Indicates a state in the data model that prevents the analysis from proceeding. Corrective action is required. Issued for items that invalidate the data model, such as: • Required data model items that are undefined • Required information that is missing

Type	Icon	Importance	Description
			• Invalid setting values (for example, selection of an invalid partitioning algorithm, duplicate data transfers on an interface, same participant on both sides of an interface) When this state is active, the **Solution** branch's **Solve** option is disabled. The **Solve** option is enabled when the error is addressed in the **Setup** branch.
		Action Optional	Indicates a state in the data model that will not prevent the analysis from proceeding, but that may be of interest. Action is possible, but not required. Issued for items that you may want to review and possibly modify before starting a solve, such as: • Output frequency set for only the last coupling step • Expected directories that do not exist and will be created if possible • Non-optimal or inappropriate setting values
		**Informational**	Provides information about the analysis that may be useful. No action is required. Informational messages may communicate details such as: • Path to the current working directory • Changes to display names in the tree
Hidden Feature	β	**Beta Feature**	When hidden features are activated and beta-level features are enabled, denotes the name and data-model location of a beta-level feature that is being used in the analysis.
	α	**Alpha Feature**	When hidden features are activated and alpha features enabled, denotes the name and data-model location of an alpha-level feature that is being used in the analysis.

System Coupling GUI Menus

This section provides quick-reference information on the menus available in the **System Coupling** GUI.

For more information, see:

Menu-Bar Menus

Context Menus

Menu-Bar Menus

The **Menu Bar** contains the **File**, **Setup**, **Solution**, **Settings**, and **Help** menus.

Available menu options for each are shown in the tables below.

Note:

File menu options cannot be used after participants have been started.

Table 15: File Menu Options

Option	Description
Open	Opens **Select working directory** dialog, which allows you to open a new working directory in the current GUI instance. By default, if multiple restart points exist, then System Coupling opens the analysis at the most recent one. Use the dialog to browse to and select the working directory to be opened. For details on the command corresponding to this operation, see `Open( )` in the *System Coupling Settings and Commands Reference* documentation.
Open at step	Enabled when a Settings file and at least one Restart file exist in the **SyC** subdirectory of System Coupling's working directory. After selecting this option, select the step where you want to open the co-simulation. System Coupling opens the analysis at the specified step. For more information, see Restarting a Coupled Analysis (p. 142). For details on the command corresponding to this operation, see `Open( )` in the *System Coupling Settings and Commands Reference* documentation.
Open at iteration	Enabled when a Settings file and at least one Restart file exist in the **SyC** subdirectory of System Coupling's working directory. After selecting this option, select the iteration where you want to open the co-simulation. System Coupling opens the analysis at the specified iteration. For more information, see Restarting a Coupled Analysis (p. 142).

Option	Description
	For details on the command corresponding to this operation, see `Open()` in the *System Coupling Settings and Commands Reference* documentation.
Open snapshot	Enabled when at least one snapshot exists in the **SyCSnapshots** directory. Opens an existing snapshot containing a previously saved state for the current co-simulation. After selecting this option, use the dialog that opens to select the snapshot you want to open the co-simulation. System Coupling locates the corresponding restart archive(s) from the **SyCSnapshots** subdirectory of its working directory. For details on the command corresponding to this operation, see `OpenSnapshot()` in the *System Coupling Settings and Commands Reference* documentation. For more information, see Working with Coupled Analysis Snapshots (p. 135).
Save	System Coupling saves a Settings file and a Restart file to the **SyC** subdirectory of its current working directory: For more information, see Saving a Coupled Analysis (p. 134). For details on the command corresponding to this operation, see `Save()` in the *System Coupling Settings and Commands Reference* documentation.
Save snapshot	Saves a snapshot of the current state of the coupled analysis. After selecting this option, specify the snapshot name in dialog that opens. System Coupling saves the snapshot to an archive file in the **SyCSnapshots** subdirectory of the working directory. For details on the command corresponding to this operation, see `SaveSnapshot()` in the *System Coupling Settings and Commands Reference* documentation. For more information, see Working with Coupled Analysis Snapshots (p. 135).
Load SCI file	Loads the specified System Coupling Input (`.sci`) file and pushes its information to the data model. For details on the command corresponding to this operation, see `ImportSystemCouplingInputFile()` in the *System Coupling Settings and Commands Reference* documentation. **Note:** The recommended method of opening a Workbench-based analysis in one of System Coupling user interfaces is to

Option	Description
	export the coupling setup. For details, see Exporting a System Coupling Setup (p. 273) and Running an Exported System Coupling Setup (p. 190).
Run script	Runs the specified Python script in System Coupling's **Command Console** tab. After selecting this option, navigate to and select the script to be run.
Start journalling	Enables engine command journalling. After selecting this option, specify path for the file to which a log of engine commands will be written as the commands are executed.
Stop journalling	Available when engine command journalling is enabled. Stops the journalling of executed commands.
Exit	Exits the co-simulation, ending the coupling run and signaling participants to end the run. This produces a clean shutdown, generating the final restart point before System Coupling disconnects from participants.

Table 16: Setup Menu Options

Option	Description
Add Participant	Opens the **Add Participant** dialog (p. 121).
Add Interface	Adds a coupling interface (p. 124) to the **Coupling Interface** branch.
Add Instancing	Adds an instancing object (p. 156) to the **Instancing** branch.
Add Named Expression	Adds a named expression (p. 73) to the **Expression** branch.
Add Reference Frame	Adds a reference frame (p. 148) to the **Reference Frame** branch.
Add Expression Function	Adds a Python expression function (p. 69) to the **Expression Function** branch.
Parallel Solve Setup	Opens the **Parallel Partitioning** dialog, which allows you to configure the participant's parallel execution settings (p. 178) for the coupled analysis.
Activate Hidden Features	Adds the **Hidden Feature Control** branch to the tree and activates hidden feature controls.

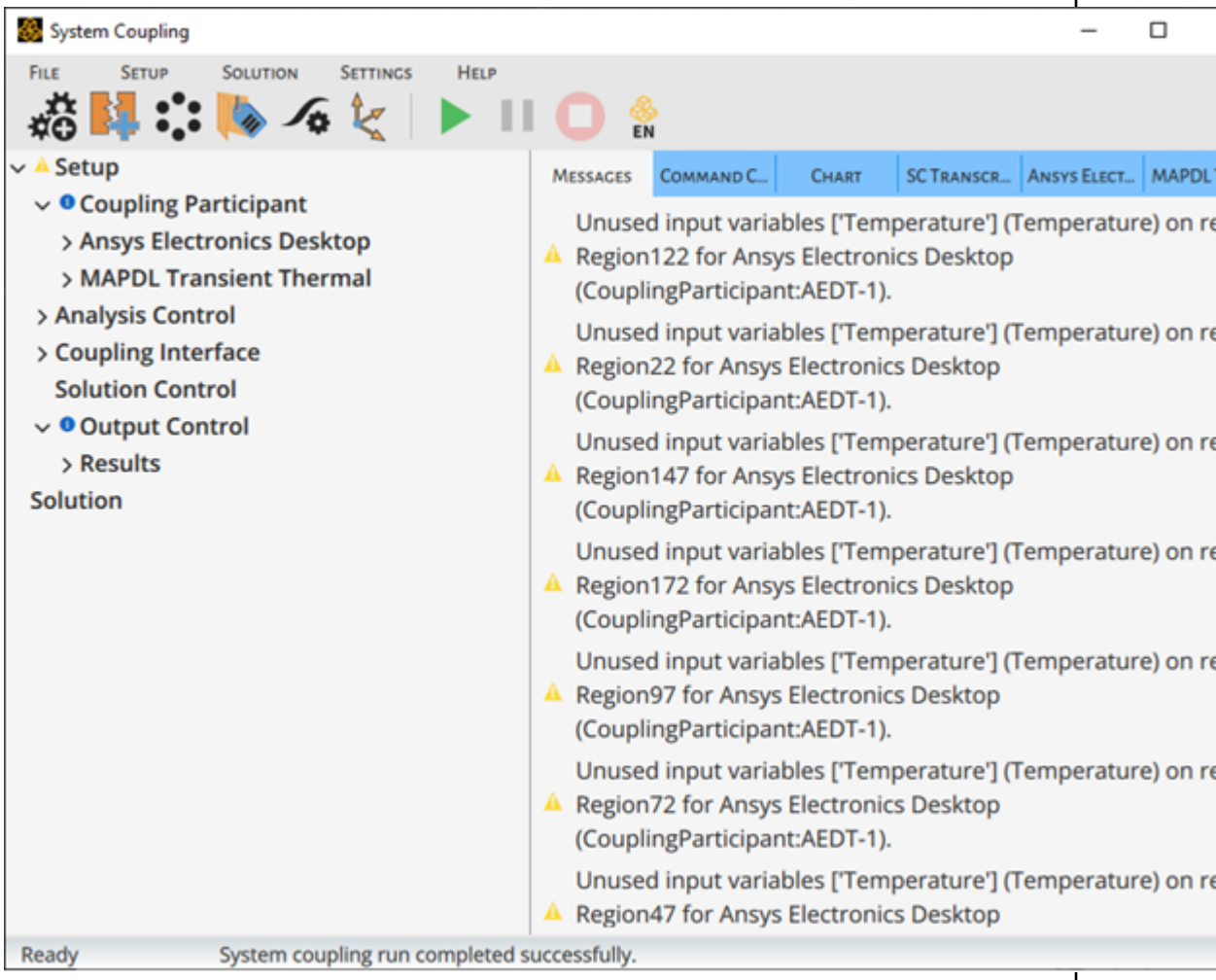
Table 17: Solution Menu Options

Option	Description
Solve	Enabled when the coupled analysis setup is free of data model errors and the solution is not currently executing. Solves the coupled analysis. (p. 132)
Interrupt	Enabled when the solution is executing.

Option	Description
	Interrupts the execution of the solution. (p. 141)
Abort	Enabled when the solution is executing. Aborts the execution of the solution. (p. 141)
Map	Enabled when the aerodamping coupled analysis setup is free of data model errors and the mapping is not currently executing. Maps the coupled analysis.
Preview Mapping	Enabled before the aerodamping coupled analysis has been mapped. Generates a mapping preview and opens it in EnSight.
Open Results in EnSight	Enabled when the solution is completed if EnSight-compatible results exist. Opens results for the current analysis in EnSight. (p. 207)

Table 18: Settings menu options

Option	Description
Switch to dark theme/Switch **Light Theme**	Sets the GUI into either the dark theme or light theme mode.

Option	Description
to light theme	 **Dark Theme**

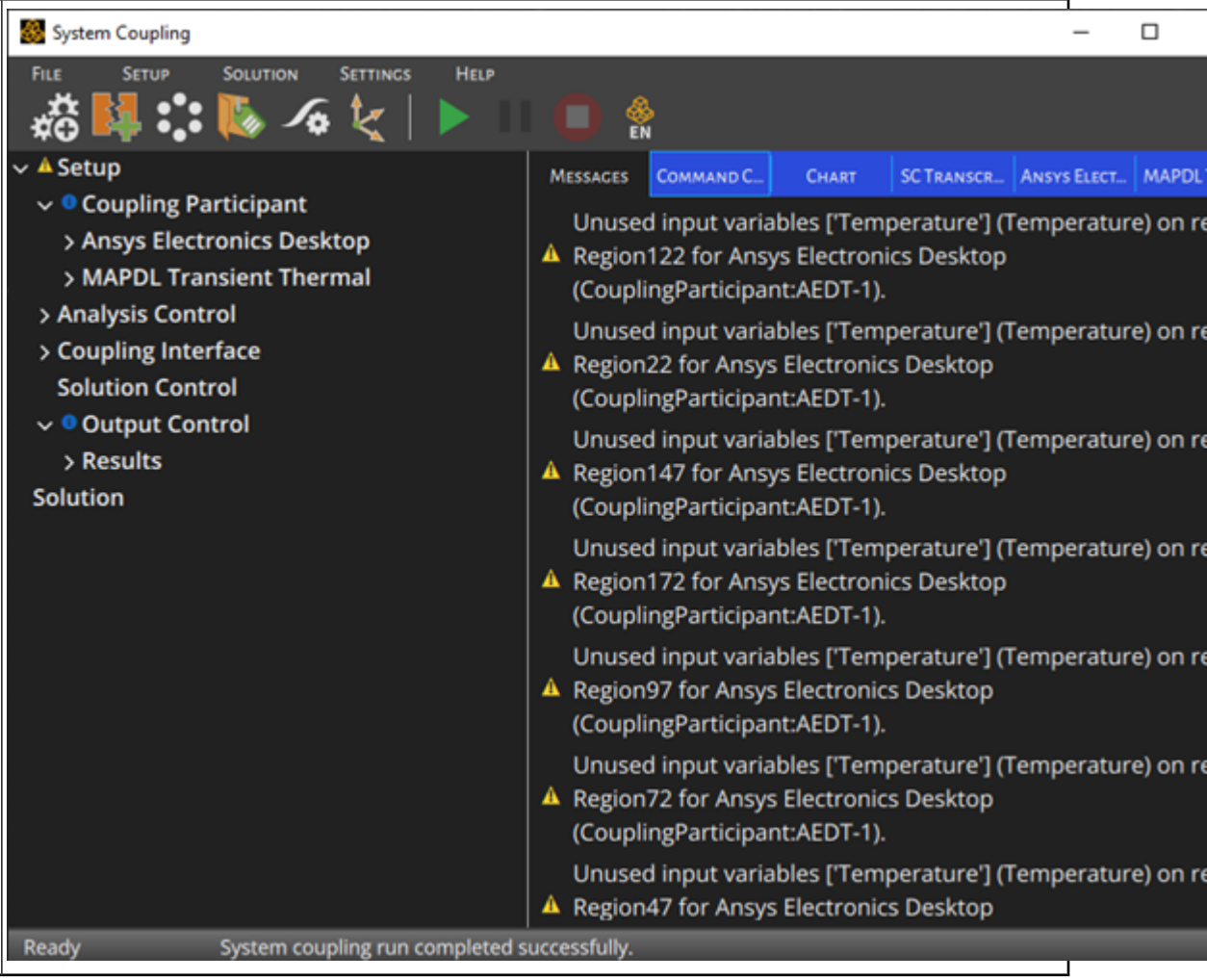
Option	Description
	

Table 19: Help menu options

Option	Description
System Coupling User's Guide	Opens the *System Coupling User's Guide*.
Settings and Commands Reference	Opens the System Coupling Settings and Commands Reference manual.
Participant Library APIs	Opens the System Coupling Participant Library documentation.
Tutorials	Opens the System Coupling Tutorials page.
Beta Features	Opens the System Coupling Beta Features documentation.
Release Notes	Opens to the System Coupling release notes page.
About	Opens a dialog containing product copyright, version, and build information.

Context Menus

When additional actions are available for an item in the **Outline** tree view, they are available as options in the corresponding context menu. Context options can be accessed by right-clicking the **Setup** branch, the **Solution** branch, and many of the items defined under them (branches, object containers, objects, and settings).

Note:

- Unless otherwise noted, the context options listed in this section are available and enabled when the specified branch/data model item exists.
- For settings with user-entered values, **Cut**, **Copy**, and **Paste** context options are usually available on the text-entry field. These options are not included in the tables that follow.

Available options are listed in the following tables:

- Table 20: General Setup context options (p. 109)
- Table 21: Setup context options (p. 109)
- Table 22: Solution and Mapping context options (p. 112)

Table 20: General Setup context options

Type	Item	Option
Object Containers	**Add**	Adds an object to the object container.
Containers	**Add <*object*>**	For singletons or object containers with child object containers, adds the specified object to the container.
Objects	**Remove**	Removes the object from the coupled analysis.
	Rename	Changes the object's display name in the tree.
Settings	**Cut**	For settings with a user-entered value, performs the selected action on the text.
	Copy	
	Paste	

Table 21: Setup context options

Tree Location	Item	Context Option	Description
Branch	**Setup**	**Add Participant**	Adds a coupling participant (p. 121) to the **Coupling Participant** branch.
		Add Coupling Interface	Adds a coupling interface (p. 124) to the

Tree Location	Item	Context Option	Description
			Coupling Interface branch.
		Activate Hidden Features	Adds the **Hidden Feature Control** branch to the tree and activates hidden feature controls.
		Add Named Expression	Adds a named expression (p. 73) to the **Expression** branch.
		Add Expression Function	Adds a Python expression function (p. 69) to the **Expression Function** branch.
		Add Reference Frame	Adds a reference frame (p. 148) to the **Reference Frame** branch.
		Add Instancing	Adds an instancing object (p. 156) to the **Instancing** branch.
Branch	**Library**	**Add Named Expression**	Adds a named expression (p. 73) to the **Expression** branch.
		Add Expression Function	Adds a Python expression function (p. 69) to the **Expression Function** branch.
		Add Reference Frame	Adds a reference frame (p. 148) to the **Reference Frame** branch.
		Add Instancing	Adds an instancing object (p. 156) to the **Instancing** branch.
Branch	**Expression Function**	**Reload Expression Function Modules**	Reloads the Python expression function modules (p. 72) into the coupled analysis.
Object	**Frame-<#>**	**Add Rotation Transformation**	Adds a rotation transformation (p. 151) to the reference frame's **Transformation** branch and the **Transformation Order** list.

Tree Location	Item	Context Option	Description
		Add Translation Transformation	Adds a translation transformation (p. 151) to the reference frame's **Transformation** branch and the **Transformation Order** list.
Branch	**Coupling Participant**	**Parallel solve setup**	Opens the **Parallel Partitioning** dialog, which allows you to configure the participant's parallel execution settings (p. 178) for the coupled analysis.
Object	**Participant-<#>**	**Generate Input File**	Available for Fluent participants when the Fluent input option is not set to **Journal File**. Generates a solver input file for the participant.
		Update	Available for server participants. Updates the participant setup (p. 383) with the changed input file.
Object	**Interface-<#>**	**Add Data Transfer**	Adds a data transfer (p. 126) to the coupling interface's **Data Transfer** branch.
		Add FSI Data Transfers	Adds available Force and Incremental Displacement transfers to the coupling interface's **Data Transfer** branch.
		Add Thermal Data Transfers	Adds available non-surface-to-surface Temperature and Heat Rate transfers to the coupling interface's **Data Transfer** branch.
		Add Thermal Data Transfers (Heat Rate)	Adds available surface to-surface Temperature and Heat Rate transfers to the coupling interface's **Data Transfer** branch.

Tree Location	Item	Context Option	Description
		Add Thermal Data Transfers (Heat Transfer Coefficient)	Adds available surface-to-surface Temperature and Heat Transfer Coefficient / Convection Reference Temperature transfers to the coupling interface's **Data Transfer** branch.
		Add Aerodamping Data Transfers	Adds selected Mode Shape data transfers (p. 377) to the coupling interface's **Data Transfer** branch.
		Add Ordered Data Transfers	Adds all possible data transfers between two Functional Mock-Up Units (FMUs) (p. 361) to the coupling interface's **Data Transfer** branch.
Branch	**Output Control**	**Add Live Visualization**	Adds a live visualization object (p. 217) under a **Live Visualization** branch.

Table 22: Solution and Mapping context options

Type	Item	Option	Description
Branch **Solution**		**Solve**	Enabled when the coupled analysis setup is free of data model errors and the solution is not currently executing. Solves the coupled analysis. (p. 132)
		Interrupt solve	Enabled when the solution is executing. Interrupts the execution of the solution. (p. 141)
		Abort solve	Enabled when the solution is executing. Aborts the execution of the solution. (p. 141)
		Open Results in En-Sight	Enabled when the solution is completed if EnSight-compatible results exist.

Type	Item	Option	Description
			Opens results for the current analysis in EnSight. (p. 207) Enabled before the aerodamping coupled analysis has been mapped.
		Preview Mapping	Generates a mapping preview and opens it in EnSight. No results are written by the CFD Server participant.
Branch	**Mapping** 1 (p. 113)	**Map**	Enabled when the aerodamping coupled analysis setup is free of data model errors and the mapping is not currently executing. Maps the coupled analysis.
		Open Results in EnSight	Enabled when EnSight-compatible results exist. Opens mapping results for the current analysis in EnSight.

1: Replaces the **Solution** branch for aerodamping analyses with one-way transfers of modal data.

Using the System Coupling Command-Line Interface (CLI)

For information on using the System Coupling command-line interface (CLI), see:

Starting the System Coupling CLI

Getting Help in the System Coupling CLI

Tip:

- System Coupling uses Python 3.10. To use the CLI effectively, you should have some familiarity with working from your operating system command prompt and a basic knowledge of the Python language. For more information, see the resources published by the Python Software Foundation.
- All sub-processes that are manually started from the command console will make use of the system environment that is setup for that console. Review and edit the environment passed to such sub-process if you encounter any problems with their execution.

Starting the System Coupling CLI

To start the System Coupling CLI:

1. Open the shell for your operating system in System Coupling's working directory.

2. Execute the command for the platform you are using:

   - **Windows**

     ```
     > "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling"
     ```

   - **Linux**

     ```
     $ "$AWP_ROOT251/SystemCoupling/bin/systemcoupling"
     ```

   The command prompt says, "`Starting Command Console...`", indicating that System Coupling has started.

Tip:

When starting System Coupling in its CLI, you can use the following command-line arguments to configure the coupling run:

- To run a script on startup, you can use the `-R` command-line option. For more information, see Command-Line Options.
- To specify how long System Coupling waits for participants to connect to the coupled analysis, use the `-C` or `--connectiontimeout` argument, as described in Setting a Participant Connection Timeout Interval (p. 158).
- If you want to start System Coupling in parallel execution mode, use the command-line parallel startup options described in Parallel Options for System Coupling (p. 184).

For a comprehensive listing of all available command-line options, see Command-Line Options in the *System Coupling Settings and Commands Reference*.

Getting Help in the System Coupling CLI

You can get help from within the System Coupling CLI using any of the following methods:

Help on Python Usage

To obtain usage information from the Python interpreter's help utility, use the `help()` command as follows:

- To access interactive help, type: **`help()`**
- To get information on an object, type: **`help(`*`object`*`)`**
- To exit the help utility, type: **`quit`**

Help on System Coupling Command-Line Options

Before starting the System Coupling CLI, you can get information on the command-line options that can be applied when System Coupling's execution script is run. To do so, you can use either the `-h` or `--help` command-line option when running the execution script. For example:

```
"%AWP_ROOT251%\SystemCoupling\bin\systemcoupling" --help
```

Descriptions of the available options will be written to the CLI, which will then close. To start System Coupling, you must run the executable again without the `-h` or `--help` option.

Available startup options are described in Command-Line Options in the *System Coupling Settings and Commands Reference*.

Help on System Coupling Commands

To obtain help on commands from within the System Coupling CLI, type in **`help(<command>)`**. For example:

```
help(Initialize)
Help on function Initialize:

Initialize(**kwds)
    Description :

        Interactive command that initializes a coupled analysis.

        This includes preparing the coupling service, establishing connections
        between the coupling service and all participants, and writing
        participant build information to the transcript.

    Supported keyword arguments :

      Essential arguments :

    Return type is NoneType
```

Available commands are described in Commands in the *System Coupling Settings and Commands Reference*.

Preparing for a Coupled Analysis

When preparing a coupled analysis to be run in one of System Coupling's user interfaces, you will perform the following steps:

Setting Up a Coupled Analysis Directory Structure

Completing Participant Setups

Additional Participant Setup Considerations

Generating Participant Files

Setting Up a Coupled Analysis Directory Structure

Set up a directory structure where you will run System Coupling and store the analysis files generated and used by System Coupling and coupling participants.

Placement of the Directory Structure

Ensure that you create the coupled analysis directory structure outside of any Workbench project directories. Even if a participant's setup is performed in Workbench, the coupled analysis is not managed by Workbench. Everything related to the coupled analysis should be stored or executed in a directory independent of Workbench processes.

Recommended Directory Structure

The recommended directory structure for a coupled analysis is as follows:

- **Coupled analysis working directory**
  - **Participant 1 working directory**
  - **Participant 2 working directory**
  - **Participant 3 working directory**

The parts of the recommended directory structure are described below:

Coupled analysis working directory:

Also called "System Coupling's working directory" or the "co-simulation working directory."

Serves as the working directory for the coupled analysis and is System Coupling's run directory.

This is where System Coupling:

- Accesses Shutdown (p. 329) (`scStop`) files
- Automatically creates the **SyC** subdirectory for its own output files:
  - Settings (p. 351) and Restart (p. 351) files
  - Log (`scLog.scl`) files
  - Any additional diagnostic or results files that may be generated
  - Automatically creates a **Results** subdirectory for its EnSight-compatible output files (p. 207).
  - If Python functions will be added for use in expressions, the manual creation of a **Modules** subdirectory is necessary.

Participant working directories:

For each participant, the coupled analysis working directory should contain a ***participant working directory*** to store coupled analysis setup, input, and output files.

Each Ansys participant should write its solver input and System Coupling Participant (`.scp`) setup files to its working directory. If you move participant files manually, be sure to place them in this directory.

This is where a participant solver:

- Writes its coupling setup file (for example, `.scp` and `.ftsim` files)
- Writes and accesses its solver input files (for example, `.cas.h5`, `.dat`, `.def`, and `.ftsim` files)
- Directs its coupling-related output, such as participant logs and transcripts
- Directs its solver-specific output and results files

This is where System Coupling:

- Accesses System Coupling Participant (`.scp`) setup files

A coupling participant's working directory is set by its `ExecutionControl | WorkingDirectory` setting in the data model. For details, see `WorkingDirectory` in the System Coupling Settings and Commands Reference documentation.

Completing Participant Setups

Finish setting up the physics and coupling-specific settings for participants to be included in the coupled analysis. For each participant, open the product and perform the following steps:

1. Complete the fluid, solid, or electromagnetic physics setup.

   During the coupled analysis, System Coupling manages participant solutions. However, you should ensure that each participant can solve successfully on its own before being included in the coupled analysis.

   Note:

   - When using same material(s) in multiple participants, take care to ensure that common properties are consistently defined.
   - When defining motion on a coupled region, ensure that the motion is defined consistently for both participants.
   - Directory paths and object names (for interfaces, participants, regions, and variables) must be in ASCII format. Object names also cannot include forward slashes (/).
   - For more information, see Additional Participant Setup Considerations (p. 118).

   For suggestions on how to ensure that your coupled analysis runs smoothly, see Building Up to a Coupled Analysis (p. 393).

2. Configure coupling-specific settings to enable coupling.

   - For CFX and Fluent, this involves creating one or more boundary conditions.

- For Forte, this involves selecting the **Enable System Coupling** check box for one or more boundary conditions.
- For Mechanical, this involves creating a **System Coupling Region**.
- For Maxwell, this involves creating a **System Coupling Setup**.

For more information on a given participant's settings, use the links in Supported Coupling Participants (p. 54) to access its documentation.

Important:

Depending on the type of analysis being performed, additional participant physics or coupling setup steps may be required. For more information, see Additional Participant Setup Considerations (p. 118) and Common Engineering Applications for System Coupling (p. 371).

Additional Participant Setup Considerations

Depending on the type of coupled analysis being performed, additional physics setup steps may be necessary. When applicable, consider:

Compatible material properties:

When using the same material(s) in multiple participants, take care to ensure that material common properties (for example, conductivity) are consistently defined. This consideration is often relevant in analyses involving thermal data transfers and, in particular, should always be taken into account for thermal analyses involving an electromagnetic participant.

For example, if Heat Rate and Temperature values are to be transferred between an electromagnetic solver and a thermal solver, both models must be made of materials that allow for temperature-dependent properties (for example, electrical conductivity and thermal conductivity, respectively).

Identify same interface sides:

For coupled analyses that include thin bodies — such as plates, baffles, airfoils, or thin fluid gaps — ensure that System Coupling can find the correct interface sides. This process is automated but can be customized, as described for the `FaceAlignment` setting in the *System Coupling Settings and Commands Reference* documentation.

Compatible topological properties:

For coupled analyses that will include data transfers between regions of different topologies, ensure that the models have compatible circumferential extents and geometric alignment. For more information, see Defining Geometry Transformations for Models with Different Orientations (p. 148).

Appropriate motion definitions:

- **Stationary bodies**: Ensure that stationary bodies will not move from the reference position or participate in multiple reference frames during the co-simulation.

- **Moving/rotating bodies**:
  - Ensure that motion/rotation is defined for all bodies that will move or rotate during the co-simulation.
  - When motion is defined on a coupled region, ensure that the motion is defined consistently for both coupling participants. (Currently, this is supported only for electromagnetic-thermal analyses between Maxwell and Fluent coupling participants.)

  Tip:

  When motion is defined on coupled bodies, Ansys recommends that you monitor the System Coupling Transcript's **Mapping Summary** sections during execution to ensure that the bodies move consistently throughout the analysis.

Aerodamping setup:

Aerodamping simulations have specific setup requirements. For details, see Aerodynamic Damping Analysis (p. 377).

Electric arc modeling setup:

Electric arc modeling simulations have specific setup requirements. For details, see Electric Arc Modeling (p. 389).

Generating Participant Files

Once participant physics and coupling setups are complete, generate the solver input file and, where applicable, the System Coupling Participant (`.scp`) setup file for each participant. The folder containing the file used to add a participant to a coupled analysis is designated as that participant's working directory, so you will generally write these files to the intended working directory for the participant.

For details on how participant working directories may be defined, see `WorkingDirectory` in the *System Coupling Settings and Commands Reference* documentation.

Participant solver input files

After you have finished each participant's setup, generate the solver input file(s) to the participant's working directory before exiting the participant application. Use the method noted in the participant's product documentation.

For details on the types of solver files supported for participants, see the following sections in the *System Coupling Settings and Commands Reference* documentation:

- Fluent participants: `FluentInput` object
- AEDT, CFX, CFD Server, Forte, Mechanical, and Mechanical Server participants: `InitialInput` setting

Participant Coupling Setup files

SCP setup files

The participant setup files needed are determined by how the participant is added (that is, with an input or executable file) and/or the participant type. If an input file is used, participant information may be populated to the data model either directly (from an `.scp` file) or indirectly (from another file format used by the participant).

- If coupling and/or automatic `.scp`-file generation are enabled for a participant, then the participant automatically generates the `.scp` file (and any other configuration files needed by System Coupling) at the same time and in the same location as its solver input file.
- If automatic `.scp`-file generation is *not* enabled for a participant, then you can generate the `.scp` file on demand using the method noted in the participant's product documentation.
- If participant `.scp` files and participant solver input files are generated when a System Coupling setup is exported from Workbench (p. 273), then these files are written to the participant working directory, which is created as a subdirectory in the co-simulation working directory specified for the export.

For details on the contents and generation of `.scp` files, see Participant Setup File (p. 311).

Other setup files

Forte, CFD Server, and Mechanical Server participants use their own setup files, as follows:

- Forte participants use a Forte solver (`.ftsim`) file.
- CFD Server participants use a CFD surface mesh (`.csv`) file.
- Mechanical Server participants use:
  - a Mechanical modal analysis results (`.rst`) file, and
  - a modal mass (`*.mode`) file, if defined for the analysis.

Creating a Coupled Analysis

To create a System Coupling analysis in either of its user interfaces, perform the following steps:

Adding the Participants

Adding a Coupling Interface

Adding a Data Transfer

Note:

For the following coupled analysis objects, display names are shown in System Coupling's GUI, Transcript/Log files, and EnSight-formatted results:

- Interfaces

- Participants
- Regions
- Variables

Adding the Participants

Each coupled analysis must have at least two coupling participants. You can add participants to the coupled analysis by specifying and loading their System Coupling Participant (`.scp`) setup files. When a participant definition loaded, the data model is populated with information from the participant's setup file.

Each participant object's display name is used, if available, and is shown in System Coupling's GUI, Transcript/Log files, and EnSight-formatted results. Otherwise, the internal object name assigned by System Coupling is used. Internally, participants are named according to the convention ***<Participant #>***, where:

"*Participant*" is the `ParticipantType`, and

"*#*" is an index indicating the order in which the participant was loaded.

For instructions on adding coupling participants in a given user interface, see:

Adding a Coupling Participant in the GUI

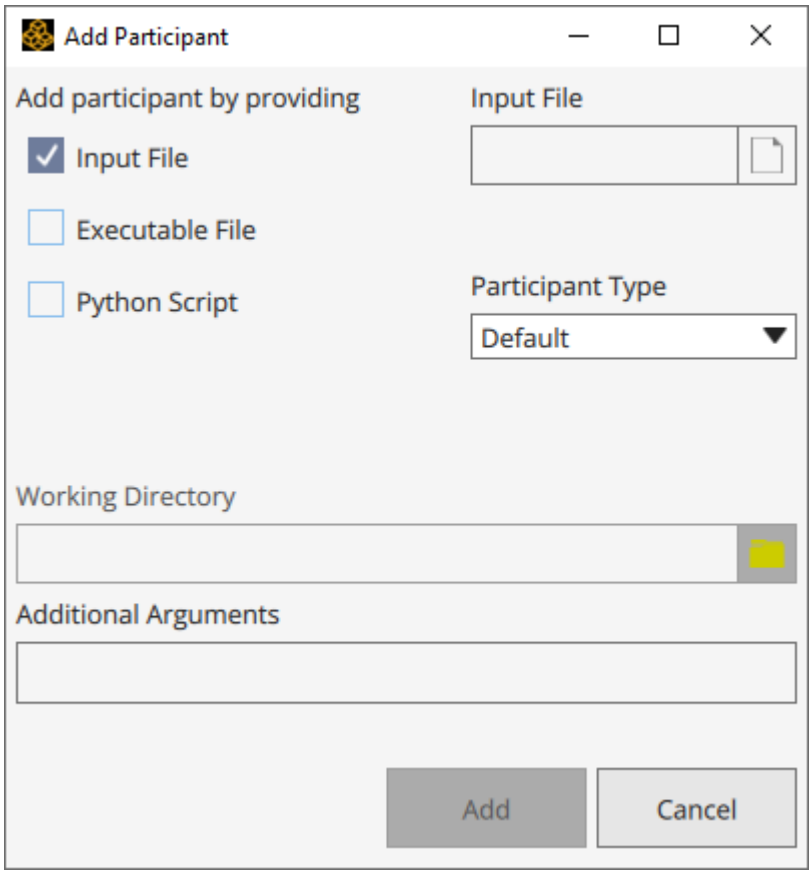
Adding a Coupling Participant in the CLI

Adding a Coupling Participant in the GUI

To add a coupling participant in the **System Coupling**, perform the following steps:

1. Perform one of the following actions:
   - From the **Menu Bar**, select **Setup** → **Add Participant**.
   - On the **Toolbar**, click  .
   - In the **Outline** tree, right-click the **Setup** branch and select **Add Participant**.

   The **Add Participant** dialog opens.



2. Select the file types you want to use to add the participant:
   - **Input File**
   - **Input File** and **Executable File**
   - **Input File** and **Python Script**
   - **Executable File**
   - **Python Script**

3. For each selected file type:
   a. Click .
   b. Browse to the file location.
   c. Click **Open**.

4. The **Participant Type** and **Working Directory** fields are populated based on the files used to add the participant. Keep these values.

   Note:

   You may change these values manually, but be aware that mismatches in participant data will result in an error and a corresponding warning written to the GUI's **Command Console** tab.

5. For **Additional Arguments**, enter any arguments to be added to the participant's executable file.

   For more information, see `AdditionalArguments (p. 185)` in the *System Coupling Settings and Commands Reference* documentation.

6. Click **Add**.

The coupling participant object is added to the data model under the **Coupling Participant** object container.

Note:

The participant analysis type cannot be changed in the GUI.

Tip:

Once the **Coupling Participant** container is available, you can right-click it and select **Add** to add another participant object.

Adding a Coupling Participant in the CLI

To add a coupling participant in the CLI, run the `AddParticipant()` command. You may add the participant by specifying an input file (using the `InputFile` argument), an executable file (using the `Executable` argument) or a Python script file (using the `PythonScript` argument). You can also specify multiple files at once (input file and executable or input file and Python script file).

Optionally, you may also specify participant information using the `ParticipantType`, `Working-Directory`, and `AdditionalArguments` arguments.

When the command is run, the coupling participant object is added to the data model under the **`CouplingParticipant`** object container.

Example 11: Add a coupling participant using an input file

```
>>> AddParticipant(InputFile = 'cfx.scp')
'CFX-1'
```

Example 12: Add a coupling participant using an executable file

```
>>> AddParticipant(Executable = 'maxwellSolver.bat')
'AEDT-1'
```

Example 13: Add a participant to a coupled analysis using a Python script

```
AddParticipant(PythonScript = "participantSolver.py")
```

Example 14: Add a participant to a coupled analysis using a Python script and an SCP file

```
AddParticipant(InputFile = "setup.scp",
               PythonScript = "participantSolver.py")
```

Adding a Coupling Interface

Each coupled analysis must have at least one coupling interface. Coupling interfaces must be added to the analysis individually.

Note:

You must create separate interfaces to transfer variable and parameter data to or from a participant.

When an interface is created, the `CouplingInterface` object container becomes available in the data model, with the coupling interface defined beneath it. When adding a coupling interface, you must specify the participant named and region(s) to be associated with each side of the coupling interface.

Interface names must be unique within the coupled analysis. When coupling interfaces are added, they are assigned default names according to the convention ***<Coupling Interface #>***, where "*#*" indicates the order in which the interfaces were created. For example, if three interfaces are created, they are named `CouplingInterface1`, `CouplingInterface2`, and `CouplingInterface3`.

For each coupling interface object and the regions defined on it, display names are used, when available, and are shown in System Coupling's GUI, Transcript/Log files, and EnSight-formatted results.

Note:

The same rules apply to the creation of interfaces in both the GUI and the CLI. For more information, see Rules for Adding Coupling Interfaces (p. 56).

For instructions on adding a coupling interface in a given user interface, see:

Adding a Coupling Interface in the GUI

Adding a Coupling Interface in the CLI

Adding a Coupling Interface in the GUI

To add a coupling interface in the **System Coupling** GUI, perform the following steps:

1. Perform one of the following actions:

   - From the **Menu Bar**, select **Setup** → **Add Interface**.
   - On the **Toolbar**, click  .
   - In the **Outline** tree, right-click the **Setup** branch and select **Add Coupling Interface**.

   The coupling interface object is added to the data model under the **Coupling Interface** object container.

   Tip:

   Once the **Coupling Interface** container is available, you can right-click it and select **Add** to add another interface object.

2. Under the coupling interface object, click **Side**.

   Two objects representing the sides of the interface, are defined under **Side**. The display name of the associated coupling participant is added to the object label.

   The settings for sides **One** and **Two** are shown in the **Properties** pane. At this point, note that the values are empty for both sides. For the data model to be valid, a different participant must be specified for each side.

3. Set the following values for each side of the interface, as follows:

   a. Select the **Coupling Participant** to be associated with the selected interface side.

   b. For **Region List**, specify the region(s) to be used in data transfers involving the selected participant, ensuring that only those regions are selected.

   Note:

   - if a participant contains at least one parameter, then the region list is available but may only be used if all data transfers (p. 126) for the interface use variables for the participant.
   - The drop-down list is populated with all the regions defined for the participant, with selected regions shown at the top.
   - By default, regions are not selected for the participant. Be sure to select at least one region.
   - A filter field at the top of the list allows you to filter the region list by entering characters and the asterisk (*) wild card as criteria. For example:

- Entering "**bus1**" might result in the following list of regions: **bus1_solid**, **bus1_shadow_wall**, **bus1_wall**.
- Entering "**bus*solid**" might result in the following list of regions: **bus1_solid**, **bus2_solid**, **bus3_solid**.

The coupling participant object is now fully defined.

Adding a Coupling Interface in the CLI

In the System Coupling CLI, the `AddInterface()` command allows you to use internal names to specify the participants and regions to be associated with each interface side.

When specifying regions, the `SideOneRegionNames` and `SideTwoRegionNames` arguments accept a list of strings indicating the names of the regions to be included on the interface.

Note:

You cannot specify regions for FMUs. If a parameter is used in any data transfer in an interface, specifying regions is not allowed on the interface side(s) with parameters.

To add a coupling interface, run the command as shown in the example below:

Example 15: Add an interface using internal participant and region names

```
>>> AddInterface(SideOneParticipantName = 'AEDT-1',
      SideOneRegionNames = 'Volume5',
      SideTwoParticipantName = ['MAPDL-2'],
      SideTwoRegionNames = ['FVIN_1'])
```

When the command is run, the coupling interface object is added to the data model under the **CouplingInterface** object container.

Adding a Data Transfer

Each coupling interface must have at least one data transfer. When one or more data transfers are created, the **Data Transfer** container object becomes available in the data model, with the data transfer object(s) defined beneath it. When adding a data transfer, you must specify the interface where the transfer is to be added, the target side for the transfer, and the variables to be associated with each side of the interface.

Data transfer names must be unique within an interface. When data transfers are added to an interface, they are assigned default names according to the convention *<TargetVariableDisplayName>-<#>*, where:

"<*TargetVariableName*>" is the display name for the target variable, as defined by System Coupling

"#" indicates the order in which the data transfers with the same target variable are created on the interface.

For each data transfer object and the variables defined on the sides of the associated interface, display names (when available) are used and are shown in System Coupling's GUI, Transcript/Log files, and EnSight-formatted results.

You may add data transfers to a coupling interface individually. If you are using the **System Coupling** GUI, you may also add groups of related data transfers at the same time. For more information, see:

Adding a Single Data Transfer

Adding a Data Transfer Group in the GUI

Note:

- The same rules apply to the creation of data transfers in both the GUI and the CLI. For more information, see Rules for Adding Data Transfers (p. 57).
- For a given type of coupled analysis, only the data transfer settings relevant to that type of analysis are shown.
- Not constructing a data transfer for an input variable or parameter could represent a setup issue. A warning is shown if input variables or parameters are not included in a data transfer.

Tip:

Data transfer settings can be adjusted to stabilize the solution and address convergence issues. For information on stabilization methods that can be applied at the data-transfer level, see:

- Quasi-Newton Stabilization Algorithm (p. 303)
- Ramping Algorithm (p. 300)
- Under-Relaxation Algorithm (p. 302)

For information on convergence criteria, see Evaluating Convergence of Data Transfers (p. 278).

Adding a Single Data Transfer

When adding a single data transfer, you must specify the interface on which the transfer is to be added, the target side for the transfer, and the variables to be associated with each side of the interface.

For instructions on adding a single data transfer, see:

Adding a Single Data Transfer in the GUI

Adding a Single Data Transfer in the CLI

Adding a Single Data Transfer in the GUI

To add a data transfer in the **System Coupling** GUI, perform the following steps:

1. In the **Outline** tree under **Coupling Interface**, locate the coupling interface object on which the data transfer is to be created.
2. Select one of the following options:
   - Right-click the interface object and select **Add Data Transfer**.
   - Select the interface object and on the **Toolbar**, click  .

   The data transfer object is added to the data model under the **Data Transfer** object container.
3. Click the data transfer object to review its settings and change, if necessary.

   The following data transfer settings are shown in the **Properties** pane, with values populated from the participant setup files:
   - **Suppress**: Allows this data transfer to be deactivated without deleting it permanently.
   - **Target Side**: Specifies the interface side that will receive the transferred data. Possible values are **One** and **Two**. Default value is **One**. The display name of the associated coupling participant is added to the object label.
   - **Source Variable**: Automatically populated with the variable for the participant associated with the source side of the interface.
   - **Target Variable**: Automatically populated with the variable for the participant associated with the target side of the interface.
   - **Ramping Option**: Specifies the ramping option to be applied. Default value is **None**.
   - **Relaxation Factor**: Specifies the relaxation factor to be applied if ramping is used. Default value is **1**.
   - **Convergence Target**: Specifies the convergence target for the data transfer. Default value is **0.01**.
4. Under the data transfer object, click **Stabilization** to review solution stabilization settings and to change if needed.

Adding a Single Data Transfer in the CLI

To add a data transfer in the System Coupling CLI, run the `AddDataTransfer()` command. This command allows you to use internal object names to specify data transfer details: the interface, the target side of the transfer, and variables to be associated with each side of the interface To add a two-way data transfer on an interface, you must run the command twice, once for each direction.

When specifying regions, use the `SourceVariable`/`TargetVariable` arguments to indicate the variable names on each side of the interface.

To add a data transfer, run the command as shown in the example below:

Example 16: Add a data transfer using internal variable names

```
>>> AddDataTransfer(Interface = 'Interface-1',
TargetSide = 'Two',
SourceVariable = 'Loss',
TargetVariable = 'HGEN')
```

When the command is run, the data transfer object is added to the data model under the `Data-Transfer` object container.

Adding a Data Transfer Group in the GUI

When working in the **System Coupling** GUI, you can add a group of related data transfers on a coupling interface. Availability of options for transfer group creation depends on the data transfers available for the specified interface.

To add a group of data transfers in the **System Coupling** GUI, perform the following steps:

1. In the **Outline** under **Coupling Interface**, locate the coupling interface object on which the data transfers are to be created.
2. Right-click the interface object or select the sub-menu to the right of the data transfer button on the **Toolbar** () and select one of the following options.
   - **Add FSI Data Transfers**: Available for Adds Force and Incremental Displacement transfers.
   - **Add Thermal Data Transfers**: Adds non-surface-to-surface Temperature and Heat Rate transfers.
   - **Add Thermal Data Transfers (Heat Rate)**: Adds surface-to-surface Temperature and Heat Rate transfers.

     If both participants can be source and target for both Temperature and Heat Rate, then System Coupling will send Heat Rate to Mechanical and Temperature from Mechanical (if available). Otherwise, the decision is arbitrary.
   - **Add Thermal Data Transfers (Heat Transfer Coefficient)**: Adds surface-to-surface Temperature and Heat Transfer Coefficient/Convection Reference Temperature transfers.
   - **Add Aerodamping Data Transfers**: Adds mode shape transfers for each selected mode shape.
   - **Add Ordered Data Transfers**: Adds all possible data transfers between two Functional Mock-Up Units (FMUs).

The relevant data transfer objects are added to the data model under the Data Transfer object container.

3. Click each data transfer object to review its settings and change them, if necessary.

   Data transfer settings relevant to the type of analysis are shown in the **Properties** pane, with values populated from the participant setup files.

   - **Suppress**: Allows this data transfer to be deactivated without deleting it permanently.
   - **Target Side**: Specifies the interface side that will receive the transferred data. Possible values are ***One*** and ***Two***. Default value is ***One***.
   - **Source Variable**: Automatically populated with an output variable for the participant associated with the interface source side.
   - **Target Variable**: Automatically populated with an input variable for the participant associated with the interface target side.
   - **Ramping Option**: Specifies the ramping option to be applied. Default value is ***None***.
   - **Relaxation Factor**: Specifies the relaxation factor to be applied if ramping is used. Default value is ***1***.
   - **Convergence Target**: Specifies the convergence target for the data transfer. Default value is ***0.01***.

4. Under the data transfer object, click **Stabilization** to review solution stabilization settings and to change them if needed.

Modifying Coupled Analysis Settings

You can modify a coupled analysis using following types of System Coupling settings:

Changing Data Model Settings

Frequently Edited Data Model Settings

Changing Data Model Settings

You can change coupled analysis settings either by using the fields in the **System Coupling** GUI or by using commands in the CLI to interact directly with the data model. For instructions on changing System Coupling's data model settings in either of its user interfaces, see:

Changing Data Model Values in the GUI

Changing Data Model Values in the CLI

Note:

For information on using expressions to define setting values, see Expressions in System Coupling (p. 59).

Changing Data Model Values in the GUI

To change the value of a data model setting in the GUI, perform the following steps:

1. In the **Outline** tree, select the branch that contains the setting.

   Each of the branch's settings is shown in the **Properties** pane, with its setting name in the left column and its value to the right.

2. In the **Value** column, change the setting value according to the type of field:

   a. Select the new value(s) from a drop-down menu. Only valid options — both in terms of value and number — are available.

   b. Type the new value into the text field. Red highlighting of a field indicates a missing value for a required setting.

Changes are applied to the data model immediately. Resulting validation warnings and errors are shown on the **Messages** and **Command Console** tabs.

Changing Data Model Values in the CLI

To change the value of a data model setting in the CLI, enter the setting's path in the CLI and use an equal sign to set the new value, as shown in the example below:

Example 17: Change the value of the TimeStepSize setting

```
>>> DatamodelRoot().SolutionControl.TimeStepSize = '1.0 [s]'
```

The change is applied to the data model when you run the command.

Frequently Edited Data Model Settings

Frequently edited data model settings are marked by an asterisk (*) in the Data Model Settings section of the *System Coupling Settings and Commands Reference* documentation.

Running a Coupled Analysis

To run a coupled analysis, perform the following steps:

Starting the Solution

Monitoring Solution Progress

Note:

System Coupling does not support an interactive mode for participants during coupled analysis runs.

Starting the Solution

When the solution is started, System Coupling connects with the coupling participants and begins the coupled analysis. A successful solution runs straight through without stopping.

For instructions on starting the solution in a given user interface, see:

Starting the Solution in the GUI

Starting the Solution in the CLI

Starting the Solution in the GUI

To start the solution in the **System Coupling** GUI, perform one of the following actions.

- From the **Menu Bar**, select **Solution** → **Solve**.
- In the **Outline** tree, right-click the **Solution** branch and select **Solve**.

Note:

This option becomes enabled only when the coupled analysis setup is completed and free of errors, including a valid working directory.

Tip:

If you wish to pause the solution at designated points during the run — for example, to review the solution's progress and adjust settings if needed — you can do this using either of the following methods:

- Use the **Solution** branch's **Interrupt solve** context option to pause execution. For more information, see Stopping a Coupled Analysis in the GUI (p. 141).
- Use System Coupling's interactive commands. For more information, see Using Interactive Solve Commands (p. 159).

Starting the Solution in the CLI

To start the solution in the CLI, run the `Solve()` command:

Example 18: Start the coupled solution

```
>>> Solve()
```

Tip:

If you wish to pause the solution at designated points during the run — for example, to review the solution's progress and adjust settings if needed — System Coupling provides

a set of interactive commands that enable you to do so. For more information, see Using Interactive Solve Commands (p. 159).

Monitoring Solution Progress

While a solution is in progress, you can monitor its progress using the following methods:

System Coupling Transcript Output

System Coupling's Transcript (p. 332) output is available in the `scLog.scl` file, the CLI console, and the GUI's **Command Console** and **SC Transcript** tabs. It provides the following information to help you to assess the progress of a coupled analysis run:

Participant Connection Details:

If any participants fail to connect, then System Coupling generates messages indicating that:

- the names of the participants that have not yet connected, and
- the time elapsed since the request for participant connections (that is, since the participants were started, or the solution was initialized or started).

These messages are written to the Transcript at the following intervals: 5 minutes, 10 minutes, 30 minutes, 1 hour, 2 hours, and 10 hours.

Tip:

When starting System Coupling from the command line, you can use the `-C` or `--connectiontimeout` argument to set the maximum amount of time System Coupling waits for participants to connect. For more information, see Setting a Participant Connection Timeout Interval (p. 158).

Solution Details:

During a coupling run, solution output is written dynamically to the Transcript. You can watch the progressive updates to its Coupled Solution (p. 345) section, either in the CLI or on the GUI's **Command Console** and **SC Transcript** tabs.

Tip:

When motion is defined on coupled bodies, Ansys recommends that you monitor the System Coupling Transcript's **Mapping Summary** sections during execution to ensure that the bodies move consistently throughout the analysis.

System Coupling Convergence Chart Data

During the execution of a coupling run, convergence data is plotted to the GUI's **Charts** tab and written to any `.csv` output files generated during the analysis. Both outputs are updated dynamically, so can be used to follow the solution as it progresses.

For more information, see Reviewing Convergence Diagnostics Charting Output (p. 197).

Participant Transcript Output

The contents of each coupling participant's transcript is written to a transcript tab in the GUI.

Common Coupled Analysis Tasks

For information on tasks that are commonly used in System Coupling workflows, see:

Saving a Coupled Analysis

System Coupling automatically saves the data model state of a coupled analysis during its creation and execution. When a coupled analysis has completed or been interrupted, the data model and a restart point file corresponding to the last completed coupling step or iteration are automatically written to HDF5-formatted files (that is, a Settings file and a Restart file) in the **SyC** subdirectory in System Coupling's working directory.

However, you can also save the state of a coupled analysis on demand, as follows:

Saving the coupled analysis state in the GUI:

- Select **File > Save** menu option.

Saving the coupled analysis state in the CLI:

- Run the `Save()` command.
- To change the output location, set the optional `FilePath` argument to the name of the intended output directory.

Note:

A *Save* operation will raise an exception if another instance of System Coupling is solving in the current working directory.

When a *Save* operation is performed, System Coupling saves the data model state to a Settings file and also saves a Restart file for each restart point that exists for the coupled analysis. These files can be used to resume the co-simulation, as described in Restarting a Coupled Analysis (p. 142).

For more detailed information on the files created, see Settings File (p. 351) and Restart Files (p. 351).

Working with Coupled Analysis Snapshots

System Coupling's snapshot capabilities allow you to capture the current state of a coupled analysis as a whole and then later restore the analysis to that state. This functionality is available for coupled analyses run using either System Coupling's GUI or its CLI.

Snapshots give you the ability to save and return to a specific point in a coupled analysis, providing you with:

- more latitude to experiment with different coupled analysis settings, without needing to remove unrelated files manually or lose existing data;
- improved portability and file management capabilities for coupled analyses, and
- reduced storage requirements when saving multiple coupled analysis configurations.

Note:

- Snapshot functionality is not available when coupling participants have been started or another instance of System Coupling is already running in the current working directory.
- Some snapshot operations can be performed only by running a command, either in System Coupling's CLI or on its GUI's **Command Console** tab.

For more information, see:

Creation of Snapshots

A snapshot captures the current state of the coupled analysis as a whole (rather than just of the data model). It includes all the files and directories contained in System Coupling's working directory that are necessary to restore the coupled analysis to its current state — specifically, the **SyC** subdirectory and the coupling working directories of all loaded coupling participants.

Both the **SyC** directory and coupling participants' working directories must be subdirectories of System Coupling's working directory. If a participant places its working directory elsewhere or attempts to reuse System Coupling's working directory, its state data will not be included in any snapshots created for the analysis. Additionally, snapshots cannot be created or opened after participants have been started (that is, when a *Solve* operation has been started (p. 131) or the `Initialize()` or `Step()` interactive solve command (p. 159) has been issued).

Snapshots can be created by either of the following methods:

Automatically:

System Coupling automatically creates an initial coupled analysis snapshot named `Initial.zip` (if it does not already exist) when the coupling participants are started by any method.

Note:

The creation of an initial snapshot is controlled by the `OutputControl.WriteInitialSnapshot` setting, which defaults to ***True***. To disable initial snapshot generation, change the value to ***False***.

On Demand:

You can create snapshots on demand using the *Save Snapshot* operation, as described in Saving a Snapshot (p. 136). There is no limit on the number of snapshots that can be created.

When a snapshot is created, System Coupling first saves the coupled analysis and its data model, then writes required state information to an archive fie (`.zip`) of the specified file name in the **SyCSnapshots** subdirectory of its working directory.

Saving a Snapshot

You can save a snapshot of the coupled analysis state at any time (provided that the coupling participants have not been started). To save an on-demand snapshot, you can use either of the following methods:

Saving a snapshot in the GUI:

Select **File > Save Snapshot** and specify a name for the snapshot in the dialog that opens.

Saving a snapshot in the CLI:

Run the `SaveSnapshot()` command, using the required `SnapshotName` argument to specify a name for the snapshot.

System Coupling writes the coupled analysis state information to an archive of the specified name in the **SyCSnapshots** subdirectory of its working directory. You can use the snapshot to restore the analysis to the saved state, as described in Opening a Snapshot (p. 137).

Overwriting an Existing Snapshot

Snapshots must have unique names within the analysis. If you attempt to save a new snapshot with the name of a snapshot that already exists, by default the existing snapshot will not be overwritten. If you wish to replace the existing snapshot with the one you are saving, use the optional `OverwriteExisting` argument with the `SaveSnapshot()` command, as follows:

```
>>> SaveSnapshot(SnapshotName='Initial', OverwriteExisting=True)
```

Printing a List of Available Snapshots

You can view a list of the snapshots available for a coupled analysis to System Coupling's Transcript. This is useful for finding the snapshot name needed to open a particular snapshot.

To do so, run the `PrintSnapshots()` command in the CLI or on the GUI's **Command Console** tab. A list containing the names and time steps of each snapshot that exists for the coupled analysis is printed to an **Available Snapshots** section in the Transcript.

```
>>> PrintSnapshots()

+--------------------------------------------------------------------------+
|                           Available Snapshots                            |
+--------------------------------------------------------------------------+
| FinalState                                                               |
|    Steps :                                                    1, 3, 5 |
| InitialState                                                             |
|    Steps :                                                               |
+--------------------------------------------------------------------------+
```

Opening a Snapshot

When directed to open a snapshot, System Coupling looks in the **SyCSnapshots** subdirectory of its current working directory for snapshot archives. You can open a snapshot using the following methods:

Opening a snapshot in the GUI:

Select the **File > Open Snapshot** menu option and use the dialog that opens to specify the name of the snapshot to be opened.

Opening a snapshot in the CLI:

Run the `OpenSnapshot()` command.

- If only one snapshot exists, then System Coupling opens that snapshot by default.
- If more than one snapshot exist, then use the required `SnapshotName` argument to specify the name of the snapshot to be opened.

System Coupling restores the coupled analysis to its previously saved state, opening the snapshot to the most recent restart point available. (Note that this is not necessarily the restart point that was open at the time the snapshot was created.) All files and directories in the snapshot are restored to the coupled analysis working directories, overwriting the files and directories corresponding to the current state. Other state files and directories (that is, which are not included in the snapshot) are not affected.

Note:

If expression functions are used in the analysis and changes were made to modules before the snapshot was saved, then inconsistencies between the modified modules and the

expression definitions may cause validation errors when the snapshot is opened later. For more information, see Reloading Expression Function Modules (p. 72).

Tip:

- To open the snapshot to a specific restart point, use the *Open* operation as described in Restarting a Coupled Analysis in an Existing GUI (Final or Intermediate Restart Point) (p. 145) or Restarting from an Intermediate Restart Point in the CLI (p. 147).
- Opening a snapshot deletes the current state of the coupled analysis working directory. If you want to retain the current state of the analysis, ensure that it has been saved to a snapshot before opening another snapshot.

Deleting a Snapshot

You can delete a snapshot from the CLI or on the GUI's **Command Console** tab.

To do so, run the `DeleteSnapshot()` command, using the required `SnapshotName` argument to specify the snapshot to be deleted.

```
>>> DeleteSnapshot(SnapshotName = 'Initial')
```

Opening a New Working Directory from the GUI

The **System Coupling** GUI provides you with the ability to switch to a new working directory without closing the GUI instance, regardless of whether a solution is already running in the current working directory. To do so, perform the following steps:

1. Select **File > Open**.

   - If there are unsaved changes in the current working directory, then System Coupling displays a dialog asking if you want to save them. Indicate whether the changes should be saved.
   - If a solution is already running in the current working directory, then System Coupling displays a dialog indicating that a run is in progress and that you can reconnect to it later (p. 139). Click **Continue** to close the dialog.

   The **Select working directory** dialog opens.

2. Navigate to the working directory to be opened, select the directory, and click **Select Folder**.

   - The specified working directory opens in the GUI. The **Command Console** tab is cleared of any content from the coupled analysis in the previous working directory.
   - If a coupled analysis has already been run in the new directory, then System Coupling restarts the analysis, as described in Restarting a Coupled Analysis in the GUI (p. 144).

- If a solution was already running in the original working directory, then it continues to run after the new working directory is opened.

Reconnecting the GUI to a Running Coupled Solution

System Coupling provides you with the ability to connect the **System Coupling** GUI to a running coupled analysis — functionality that can be especially useful when a solution is expected to have a long runtime. This feature allows you to start the analysis, disengage from it, and then come back later to connect/reconnect to the running instance of System Coupling and check on the simulation's progress.

For more information, see:

Prerequisites

Disconnecting from the Running Solution

Reconnecting to the Running Solution

Prerequisites

To connect/reconnect the **System Coupling** GUI to an analysis in-progress, the instance of System Coupling executing the analysis must meet the following criteria:

System Coupling is running in GUI Server mode.

System Coupling must be running in GUI Server mode, a GUI-only mode in which System Coupling is launched as a server and waits for an instance of the **System Coupling** GUI to connect to it.

System Coupling is always started in this mode when opened in its GUI. It can also be started in GUI Server mode from the command line, by appending the `--guiserver` argument to its executable. Note that when started in GUI Server mode from the command line, System Coupling operates in a non-interactive session (that is, its CLI does not stay open after execution of a script).

When started in GUI Server mode, System Coupling writes a GUI Server file (p. 331) to its working directory. This file contains the server port and host information needed to connect/reconnect an instance of the GUI to the corresponding coupling solution while it is still progressing.

System Coupling's working directory is visible to the GUI.

The System Coupling instance may be running either locally or on a remote host. If running on a remote host, then its working directory must be visible/accessible to the GUI attempting to connect.

System Coupling has no active GUI connections in the working directory.

System Coupling must have no active connection with an existing instance of the **System Coupling** GUI that is running in the working directory. The presence of a GUI Server file (p. 331) in the working directory indicates that another connection already exists.

Disconnecting from the Running Solution

To reconnect an instance of the **System Coupling** GUI to a coupled analysis that is in progress, the solution must be actively running with System Coupling in GUI Server mode, and it must be disconnected from any GUI. To do this, use one of the following methods:

Disconnecting from the Solution in the GUI

To start and disconnect from the solution using the **System Coupling** GUI, perform the following steps:

1. Open the **System Coupling** GUI as described in Starting the System Coupling GUI (p. 87).

   When opened in its GUI by any method, System Coupling always starts in GUI Server mode.

2. Set up the analysis and start its solution.

3. Disconnect the GUI from the running analysis, as follows:

   a. Select **File > Exit**.

      A dialog opens, indicating that a solution is in progress and that you may reconnect to it later.

   b. Click **Continue**.

The GUI closes, but System Coupling continues to run and execute the coupled analysis.

Disconnecting from the Solution from the Command Line

To start and disconnect from the solution using the command line, perform the following steps:

1. Open the shell for your operating system in System Coupling's working directory.

2. Run System Coupling's executable, using the `--guiserver` argument to start in GUI Server mode and the `-R` argument to run a coupling script.

   - **Windows**

     ```
     > "%AWP_ROOT195%\SystemCoupling\bin\systemcoupling" -R <scriptname> --guiserver
     ```

   - **Linux**

     ```
     $ "$AWP_ROOT251/SystemCoupling/bin/systemcoupling" -R <scriptname> --guiserver
     ```

This starts System Coupling in GUI Server mode, creates the analysis, and begins the solution.

Note that System Coupling is running in a non-interactive mode, which means that its CLI closes after the script is run. The only way to interact with an analysis started using this method is to connect it to an instance of the **System Coupling** GUI during its execution.

Reconnecting to the Running Solution

To connect/reconnect to a solution in progress, open the **System Coupling** GUI in the working directory for the coupled analysis.

- If the working directory already contains a `sycGui.srv` file, then the GUI will use the server port and host information it contains to connect with the solution in progress.

  When the GUI connects to the running analysis, System Coupling's connection and execution details are written to the Transcript output in the **Command Console** tab. Solution details are written to the Transcript during the remainder of its execution, allowing you to monitor its progress.

- If the directory does not contain a `sycGui.srv` file (for example, System Coupling may have exited when the solution completed), then the **System Coupling** GUI opens there, starting System Coupling in GUI Server mode and generating its own `sycGui.srv` file

Connection to an analysis requires that System Coupling is still running in the working directory. System Coupling exits under the following circumstances:

- System Coupling is both idle and not attached to a GUI instance — for example, when:
  - the GUI is exited when no coupling processes are running, or
  - the solution completes after the GUI has disconnected.
- An *Interrupt* or *Abort* operation has been requested for the analysis.

For details on issues you may encounter when using System Coupling's reconnect capabilities, see Known Issues and Limitations (p. 47).

Stopping a Coupled Analysis

For instructions on stopping a coupled analysis in a given user interface, see:

Stopping a Coupled Analysis in the GUI

Stopping a Coupled Analysis in the CLI

Stopping a Coupled Analysis in the GUI

When using the **System Coupling** GUI, right-click the **Solution** branch of the **Outline** tree, or open the **Solution** menu from the **Menu Bar**, and select one of the following options:

Interrupt solve

When the **Interrupt solve** option is selected, the coupled analysis is stopped and System Coupling ends participant processes with a clean shutdown — that is, the run continues until the current

coupling step (or the current coupling iteration, for iteration-based steady analyses) is completed and System Coupling generates a corresponding restart point and Restart file before disconnecting.

Abort solve

When the **Abort solve** option is selected, the coupled analysis does not shut down cleanly — that is, System Coupling stops the run immediately without producing any restart data and signals participants that they should shut down, as well.

When the solution stops, the Solution Lock file (p. 331) is removed from the working directory. When System Coupling exits, the GUI Server file (p. 331) is removed from the working directory.

Stopping a Coupled Analysis in the CLI

When using the System Coupling CLI, you can stop a coupled analysis using the methods described below:

Run the Shutdown() command.

When using an interactive solve process described in Using Interactive Solve Commands (p. 159), you can use the `Shutdown()` command to stop down the analysis at any point at which the solution is paused. This produces a clean shutdown, generating a restart point and writing the corresponding Restart file before System Coupling is disconnected.

Use a Shutdown file.

The ***Shutdown*** file (p. 329) is a text file named `scStop` that indicates whether the coupling run should be interrupted or aborted. When System Coupling finds the file in its working directory, it shuts down as soon as possible according to the contents of the file and signals participants that they should shut down, as well.

- For command-line runs, you must create the Shutdown file and place it in System Coupling's working directory manually.
- When a Shutdown file is used, System Coupling ends participant processes for participants that have been started automatically.

When the solution stops, the Solution Lock file (p. 331) is removed from the working directory. If System Coupling was run in GUI Server mode, the GUI Server file (p. 331) is removed from the working directory when System Coupling exits.

Restarting a Coupled Analysis

You can restart a completed or interrupted coupled analysis from either the final restart point or an intermediate restart point. Based on the restart point selected, System Coupling automatically finds all the necessary restart files (both for itself and for all participants) and then restarts the run from the designated point in the analysis. Details for the restarted run are appended to the end of the existing `scLog.scl` file.

Note that some participants may not allow restarts. For example, Mechanical participants do not support restarts for steady-state thermal and transient thermal analyses.

If any participant does not support restarts, then restarts are disabled for the coupled analysis. A participant's restart capabilities may vary according to the type of analysis being performed.

Restarts from the final result

Restarts from the final restart point are possible only with analyses that have been completed or interrupted (not aborted).

If restarting from a final restart point, you must extend the analysis to perform the restart.

Restarts from an intermediate result

Restarts from an intermediate restart point are possible with any analysis for which intermediate start points have been generated. An aborted run can be restarted from an intermediate restart point that was completed before the analysis was aborted.

When an analysis is restarted from an intermediate restart point, all subsequent restart points are removed.

- To keep restart data for one of these later coupling steps or iterations, save the analysis state to a snapshot before starting participants. For more information, see Working with Coupled Analysis Snapshots (p. 135).
- If the files for subsequent restart points cannot be removed — for example, if restart points from the previous run are currently being viewed in EnSight or another postprocessing application — then System Coupling generates an exception message instructing you to free up the files and attempt the restart again.

To return to the beginning of an analysis and start over, the recommended method is to use the initial snapshot created by System Coupling, as described in Working with Coupled Analysis Snapshots (p. 135). Alternatively, you can delete all the Restart files. In this case, however, you may need to edit settings to return the data model to its original state because the Settings file contains the values used for the most recent run.

Tip:

Editable restart scripts may be generated for Mechanical and Fluent coupling participants. For details, see:

- Using Custom Restart Files in System Coupling in the *Mechanical User's Guide*
- Using Using Custom Input Files in System Coupling's User Interfaces in the *Fluent User's Guide*

For more information, see:

Creation of Restart Points

Restarting a Coupled Analysis in the GUI

Restarting a Coupled Analysis in the CLI

Creation of Restart Points

Restarts of a coupled analysis require corresponding restart points in System Coupling and in each of the coupling participants. Restart points are generated at the end of a coupling step or coupling iteration. A new restart point can be created by the following methods:

- When requested by:
  - `OutputControl` data model settings, or
  - the `CreateRestartPoint()` command.
- When the analysis is shut down cleanly because:
  - the solution has completed;
  - the solution was interrupted by the GUI's **Interrupt solve** menu option or a `Shutdown (p. 329)` file, or
  - the analysis was stopped by the `Shutdown()` command.

For each restart point created, the Settings (p. 351) file is updated and a Restart (p. 351) file is generated. These files contain the information necessary to resume the analysis at the designated point.

Tip:

The commands noted above can be used both in the CLI and on the GUI's **Command Console** tab. For more information, see Using Interactive Solve Commands (p. 159).

Restarting a Coupled Analysis in the GUI

When restarting a coupled analysis in the **System Coupling** GUI, you can restart from either the final restart point or an intermediate restart point. The steps vary according to whether the analysis is being restarted by opening a new instance of the GUI or by specifying restart details in an existing instance of the GUI. For more information, see:

Restarting a Coupled Analysis in a New GUI (Final Restart Point)

Restarting a Coupled Analysis in an Existing GUI (Final or Intermediate Restart Point)

Restarting a Coupled Analysis in a New GUI (Final Restart Point)

When you start the **System Coupling** GUI in a working directory in which the state of a previous coupling run has been saved (that is, Restart files (p. 351) have been written to the **SyC** subdirectory), System Coupling automatically loads that analysis in its most recent state. Based on the state of the analysis, the restart may occur from either the final or an intermediate restart point.

To restart a coupled analysis by opening a new instance of the **System Coupling** GUI, perform the following steps:

1. Open the **System Coupling** GUI using one of the methods described in Starting the System Coupling GUI (p. 87).

2. On the **Select working directory** dialog, navigate to the working directory of the coupled analysis to be restarted, select the directory, and click **Select Folder**.

   The coupled analysis opens at the end of the last coupling step or iteration completed in the previous run and loads all information (both for System Coupling and participants) required to restart from that point.

   If this point is also the final step or iteration defined for the analysis (that is, the solution was completed in the previous run), then the Transcript indicates that the analysis must be extended in order to be restarted.

3. If applicable, extend the analysis using the **Number of Steps** or **End Time** setting (defined under **Solution Control**).

4. If desired, make any additional changes to analysis settings for System Coupling and/or participants.

   Changes saved prior to the restart (either in System Coupling or in any participant except Mechanical) are written to the Settings file and included in the restarted run.

5. Right-click the **Solution** branch and select **Solve**.

System Coupling automatically starts all participants from the selected coupling step or iteration and continues the solution with the next step or iteration.

Restarting a Coupled Analysis in an Existing GUI (Final or Intermediate Restart Point)

In an instance of the **System Coupling** GUI that is already open, you can restart an analysis from either a final or an intermediate restart point. To do so, perform the following steps:

1. Select **File > Open at step** or **File > Open at iteration**, depending on the type of Restart files generated for the analysis.

   Tip:

   - By default, the dialog opens to the **SyC** working directory. If the Restart files are saved to a non-default location, then navigate to that directory.
   - Restart files may have been generated for either coupling steps or coupling iterations, based on the type of the analysis and whether the participants involved support the creation of restart points per iteration. For more information, see Restart Files (p. 351).

2. In the dialog that opens, select the Restart file from which you want to restart the analysis and click **Open**. (To start from the final restart point, select the most recent file created.)

The coupled analysis opens at the end of the specified step or iteration. If the final step or iteration defined for the analysis has completed, then the Transcript indicates that the analysis must be extended in order to be restarted.

Note:

If the files for subsequent restart points cannot be removed, then System Coupling generates an exception message instructing you to free up the files and reattempt the **Open at step** or **Open at iteration** operation.

3. If applicable, extend the analysis using the **Number of Steps** or **End Time** setting (defined under **Solution Control**).
4. If desired, make any additional changes to analysis settings for System Coupling and/or participants.

   Changes saved prior to the restart (either in System Coupling or in any participant except Mechanical) are written to the Settings file and included in the restarted run.
5. Right-click the **Solution** branch and select **Solve**.

System Coupling automatically starts all participants from the selected coupling step or iteration and continues the solution with the next step or iteration.

Tip:

Alternatively, you can run the solution interactively by issuing commands on the **Command Console** tab. For more information, see Using Interactive Solve Commands (p. 159).

Restarting a Coupled Analysis in the CLI

When restarting a coupled analysis in the System Coupling CLI, the steps vary according to whether the analysis is being restarted from final or an intermediate restart point. For more information, see:

Restarting from the Final Restart Point in the CLI

Restarting from an Intermediate Restart Point in the CLI

Restarting from the Final Restart Point in the CLI

To restart a coupled analysis from the final restart point in the System Coupling CLI, perform the following steps:

1. Run the `Open()` command without arguments.

Tip:

If your Restart files are not saved to the default **SyC** directory, then use the optional `FilePath` argument to specify the directory.

By default, System Coupling finds the last coupling step or iteration completed in the previous run and loads all information (both for System Coupling and participants) required to restart from that step or iteration.

2. If the previous run completed, extend the analysis using the `NumberOfSteps` or `EndTime` setting (defined under `SolutionControl`).

3. If desired, make any necessary changes to analysis settings for System Coupling and/or participants.

   Changes saved prior to the restart (either in System Coupling or in any participant except Mechanical) are written to the Settings file and included in the restarted run.

4. Start the solution using your preferred solve process (using either the `Solve()` command or interactive solve commands (p. 159)).

System Coupling automatically starts all participants from the last completed coupling step or iteration and continues the solution with the next step or iteration.

Restarting from an Intermediate Restart Point in the CLI

To restart a coupled analysis from an intermediate restart point in the System Coupling CLI:

1. Run the `Open()` command with the `CouplingStep` or `CouplingIteration` argument set to the number of the step or iteration from which you want to restart the analysis.

   Tip:

   - By default, the dialog opens to the **SyC** working directory. If your Restart files are not saved to the default **SyC** directory, then use the optional `FilePath` argument to specify the directory.
   - Restart files may have been generated for either coupling steps or coupling iterations, based on the type of the analysis and whether the participants involved support the creation of restart points per iteration. For more information, see Restart Files (p. 351).

   System Coupling automatically finds the specified coupling step or iteration and loads all information (both for itself and participants) required to restart from that step or iteration.

   Note:

   If the files for subsequent restart points cannot be removed, then System Coupling generates an exception message instructing you to free up the files and reissue the `Open()` command with the `CouplingStep` or `CouplingIteration` argument.

2. If desired, make any necessary changes to analysis settings for System Coupling and/or participants.

Changes saved prior to the restart (either in System Coupling or in any participant except Mechanical) are written to the Settings file and included in the restarted run.

3. Start the solution using your preferred solve process (using either the `Solve()` command or interactive solve commands (p. 159)).

System Coupling automatically starts all participants from the specified coupling step or iteration and continues the solution with the next step or iteration.

Advanced Coupled Analysis Tasks

For information on tasks that may be used in advanced System Coupling workflows, see:

Defining Geometry Transformations for Models with Different Orientations

Defining Interface Instancing for Cylindrical Geometry Models

Setting a Participant Connection Timeout Interval

Using Interactive Solve Commands

Simultaneous Execution of Participant Solutions

Using Parallel Processing Capabilities

Running an Exported System Coupling Setup

Requesting Debugging and Diagnostic Output

Defining Geometry Transformations for Models with Different Orientations

System Coupling supports the creation of interfaces and data transfers between source and target models with different orientations. In a coupled analysis involving models with different orientations, the participant geometries must be aligned closely enough to allow for accurate mapping between the sides of the coupling interface (for example, the rotor and stator in an electric motor simulation are in the same relative position).

To move the geometries into the same reference orientation, use System Coupling's **reference frame** functionality to define reference frames and transformations in the `Library` and then apply the transformations to one or more coupling interface sides.

The **global reference frame** is the base system for the coupled analysis — that is, all transformations in the analysis are performed relative to this initial reference frame.

Multiple levels of nested reference frames may be created. Additional frames may be nested under these, allowing for multiple levels of parent-child relationships.

When defining reference frames:

- Reference frames and transformations must be defined prior to analysis initialization. If a modified after initialization, then the changes will not be reflected in the analysis.

- Each reference frame must have at least one transformation defined for it.

  Note:

  Rotation transformations have a default unit of radians.

- Each reference frame should be exercised in the analysis, either defined for a coupling interface side, used as a parent reference frame, or used in an instancing object.
- When `ReferenceFrame.Option` is set to ***`ByTransformation`***, each transformation defined must be added to the reference frame's `TransformationOrder` list.

  Note:

  When postprocessing coupling results in EnSight, a coupling region can be shown in only one orientation. If a region is involved in multiple coupling interfaces with dissimilar transformations, then it is shown in the `GlobalReferenceFrame`. To avoid region-orientation disparities in postprocessing output, Ansys recommends that you set up the analysis such that participant regions are transformed into consistent reference frames. For example, you might define the reference frame orientation of each coupling interface such that the geometries are in their actual real-world physical orientation.

For more information, see:

Adding Reference Frames

Adding Transformations

Using Automatic Interface Alignment

Defining Transformations for a Coupling Interface Side

Adding Reference Frames

You can add reference frames using either the GUI or the CLI. Using the steps indicated below, you may create multiple reference frames as needed, optionally using the `ParentReferenceFrame` setting to create a hierarchy of nested frames.

Note:

Reference frames are not available if at least one side of an interface is an FMU participant.

When you add the first reference frame object, the `Library` singleton is added to the data model (if it does not already exist), with the new `ReferenceFrame` object defined beneath it.

For more detailed information on reference frame definition requirements, see the `ReferenceFrame` data model object and the `AddReferenceFrame()` command in the *System Coupling Settings and Commands Reference* documentation.

Adding a Reference Frame in the GUI

1. Perform one of the following actions:

- From the **Menu Bar**, select **Setup** → **Add Reference Frame**.
- In the **Outline** tree, right-click the **Setup** branch and select **Add Reference Frame**.

The **Reference Frame** branch is added to the **Library** branch, with the new reference frame defined underneath it.

Tip:

Once the **Reference Frame** branch exists, you may add new reference frames by right-clicking it and selecting **Add**.

2. Click the reference frame object.

   Corresponding settings are shown in the **Properties** pane below.

3. Optionally, you may set a parent frame for the added reference frame.

   The **Parent Reference Frame** setting defaults to **GlobalReferenceFrame**. If other valid reference frames have been defined, you may select one from the drop-down menu.

Adding a Reference Frame in the CLI

Run the `AddReferenceFrame()` command. By default, the frame is created as a child of the global reference frame.

Optionally, you can use the `ParentReferenceFrame` argument to specify the name of a valid reference frame that will be parent to the frame created.

```
AddReferenceFrame(ParentReferenceFrame = 'Frame-1')
'Frame-2'
```

When the command is run, the `ReferenceFrame` object is defined under the `Library` singleton.

Ensure that each reference frame added has at least one transformation defined for it and is exercised in the analysis.

Note:

- By default, `Option` is set to ***`ByTransformation`***.
- The `TransformationOrder` setting is available when `Option` is set to ***`ByTransformation`***. This setting is blank when the reference frame is first added. It is automatically populated when transformations are defined by using the **Add Transformation** GUI option or running the `AddTransformation()` command. Transformations are listed in the order they are added. You may edit this list if transformations are not added in the order they should be executed.

Adding Transformations

When `ReferenceFrame | Option` is set to *`ByTransformation`*, can add transformations using either the GUI or the CLI. Multiple transformations can be added to a reference frame. When possible, transformations should be added to a reference frame in the order they are to be executed in.

When you add the `Transformation` object, it is defined under the parent `ReferenceFrame` object.

Additional arguments are required to define each type of transformation. For details on the arguments needed for each transformation type, see the `Transformation` data model object and the `AddTransformation()` command in the *System Coupling Settings and Commands Reference* documentation.

Adding a Transformation in the GUI

1. In the **Outline** tree under **Library**, right-click the reference frame to which the transformation will be added and select **Add Rotation** or **Add Translation**.

   Note:

   Rotation transformations have a default unit of radians.

   The **Transformation** branch is added to the tree, with the new transformation defined underneath it.

   Note:

   The transformation type cannot be changed for transformation objects added in the GUI.

   Tip:

   Once the **Transformation** branch exists, you may add new transformations by right-clicking it and selecting **Add Rotation** or **Add Translation**.

2. Click the transformation object.

   Settings corresponding to the transformation type are shown in the **Properties** pane below.

3. Set the additional parameters needed to define the specified transformation type.

Adding a Transformation in the CLI

Run the `AddTransformation()` command, using the required `ReferenceFrame` argument to specify the frame to which the transform will be added, the `TransformationType` argu-

ment to specify the type of transformation (*Translation* or *Rotation*), and any additional arguments needed to define the specified transform type.

Note:

Rotation transformations have a default unit of radians.

```
AddTransformation(
 ReferenceFrame='Frame-1',
 TransformationType='Rotation',
 Axis='XAxis',
 Angle='45 [degrees]')

'Rotation-1'
```

When the command is run, the `Transformation` object container is defined under the parent reference frame object.

Using Automatic Interface Alignment

If participant models are misaligned, for cases with surface-to-surface mapping you can use System Coupling's automatic interface alignment feature instead of applying transformations manually. Automatic alignment transformations can be applied to one side of an interface.

To enable auto-alignment, set the `Library.ReferenceFrame.Option` setting to *Automatic*.

When automatic alignment is enabled, System Coupling uses rigid-body motion (translation and rotation without any bending or stretching) to align the source and target geometries automatically, moving the surfaces as rigid bodies until the sides of the interface match. Source-side transformations are always applied first.

Note that automatically defined transformations can be applied only once. If more than one interface must be automatically aligned or a single interface requires multiple automatic alignments, then you must create more than one `ReferenceFrame` objects with `Option` set to ***Automatic***.

If there is a point that is duplicated in multiple instances, automatic alignment may not work, especially if the total number of points is relatively small.

Geometries with axial symmetry cannot be fully aligned, but may be partially aligned - specifically, the centroid and normal axes can be aligned.

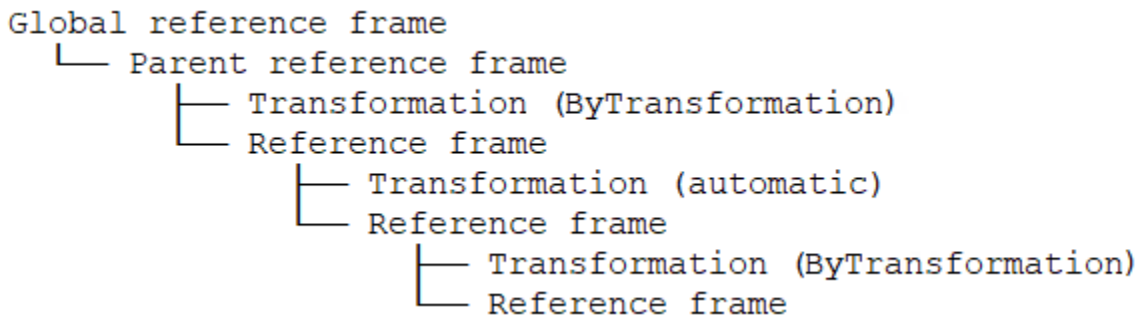
Automatic alignment for point cloud to volume, surface, or point cloud is available when beta features are activated.

Chaining Transformations

If the geometries are significantly misaligned, you may need to apply one or more "manual" transformations (ones defined using the `ReferenceFrame.Option.ByTransformation` option) in conjunction with transformations defined using auto-alignment. For example, a large enough difference between the interfaces could cause auto-alignment to align them incorrectly

(backward or upside down). It may be necessary to pre-apply a manual rotation so that the geometries align with the correct orientation..

To chain transformations, you would perform a manually defined transformation before a transformation defined using automatic alignment. If further adjustment is needed, then you could follow the automatic alignment transformation with another manually defined transformation, as follows:



Note that you may add multiple transformations of either type in sequence, as needed. For example, you could add several manually defined transformations either before or after the automatic transformation, and you could also add several automatic alignment transformations in a row.

For information on chaining transformations, see:

Chaining Manual-Automatic-Manual Transformations in the GUI

Note:

When supplying a parent `ReferenceFrame` to a `ReferenceFrame` with `Option` set to ***`Automatic`***, setting the parent `ReferenceFrame.Transformation.Option` to ***`UserDefined`*** yields an incorrect transformation.

To avoid this issue, set the parent reference frame's `Option` to ***`XAxis`***, ***`YAxis`***, or ***`ZAxis`***.

---

1. Define the first transformation created via the ***`ByTransformation`*** option.

   a. Under **Setup**, right-click **Library** and select **Add Reference Frame**.

      The **Reference Frame** branch is added, with **Frame-1** defined beneath it.

   b. Right-click **Frame-1** and select one of the **Add Transformation** options.

      A transformation is added.

   c. Use the **Transformation Order** setting to specify the order in which the transformation will be executed.

2. Define the automatic alignment.

a. Right-click the **Reference Frame** branch and select **Add**.

   Frame-2 is added.

b. For **Frame-2**, set the following properties:

   - Set the **Option** to **Automatic**.
   - Set **Parent Reference Frame** to **Frame-1**.

3. Define another transformation created via the ***ByTransformation*** option.

   a. Right-click the **Reference Frame** branch and select **Add**.

      Frame-3 is added.

   b. For **Frame-3**, set **Parent Reference Frame** to **Frame-2**.

   c. Right-click **Frame-3** and select one of the **Add Transformation** options.

      A transformation is added.

   d. Use the **Transformation Order** setting to specify the order in which the transformation will be executed.

Chaining Manual-Automatic-Manual Transformations in the CLI

```
frame1 = DatamodelRoot().Library.ReferenceFrame["Frame-1"]
frame1.Option = "ByTransformation"
frame1.ParentReferenceFrame = "GlobalReferenceFrame"
t1 = frame1.Transformation["Transformation-1"]
t1.Option = "Rotation"
t1.Axis = "XAxis"
t1.Angle = "45 [deg]"
frame1.TransformationOrder = ["Transformation-1"]

frame2 = DatamodelRoot().Library.ReferenceFrame["Frame-2"]
frame2.Option = "Automatic"
frame2.ParentReferenceFrame = "Frame-1"

frame3 = DatamodelRoot().Library.ReferenceFrame["Frame-3"]
frame3.Option = "ByTransformation"
frame3.ParentReferenceFrame = "Frame-2"
t2 = frame3.Transformation["Transformation-1"]
t2.Option = "Translation"
t2.Vector = [0.0, 0.0, 0.5]
frame3.TransformationOrder = ["Transformation-1"]
```

Defining Transformations for a Coupling Interface Side

Each side of a coupling interface has a `ReferenceFrame` setting that allows you to specify the reference frame from which any geometry transformations will be performed. By default, it is set

to the global reference frame. If other reference frames have been defined, they are populated to and can be selected from the setting's drop-down menu.

Note:

In most cases, a reference frame does not need to be set for both the interface side and the instancing object associated with that side. This could indicate an error in the interface side setup.

If transformations were not added to the frame in the correct order of execution and `ReferenceFrame.Option` is set to *`ByTransformation`*, you can reorder the list for the frame's `TransformationOrder` setting.

When the analysis is solved, reference frames are executed hierarchically from the top down, with parent frames before child frames. Within a given reference frame, transformations are executed in the order specified by the `TransformationOrder` setting.

The composite transformation is printed in the Reference Frame and Instancing Information (p. 343) section of the Transcript. When the solve is complete, it can be queried using the `GetTransformation()` command.

Defining Interface Instancing for Cylindrical Geometry Models

System Coupling supports instancing for cylindrical geometry models, which allows for the full geometry to be represented by replicating a subset of the geometry multiple times. This functionality is available for all supported topologies and is useful for applications such as gas turbine FSI problems and thermal-electromagnetic electric motor studies.

To use System Coupling's instancing functionality, start by adding `Instancing` objects in the `Library`. When adding an instancing object, use the available settings to define the rotation point, axis of rotation, and the number of instances in a full circle.

Before analysis initialization, define the instancing object for a coupling interface side. If instances are modified after initialization, then the changes will not be reflected in the analysis. Ensure that each instancing object is exercised in the analysis — that is, define each instancing object for at least one coupling interface side.

Once you have started the solution, System Coupling constructs the instances after the geometries have been aligned. Each replicated interface side receives the same data as the participant interface side being replicated (for example, replicated regions cannot have different values, scaling, and so on in a modal analysis. For vector data, any required transformation will be applied.

Solution details are written to the Transcript in terms of the full meshes — that is, the reference instance (with the participant mesh), plus any additional instances (with transformed meshes).

By default, only the reference instance (with the original mesh) is included in System Coupling's postprocessing output. You may include all instances (with all mesh transformations) using the `IncludeInstances` output control setting.

If the *Initialize* operation was issued (either explicitly by running the `Initialize()` command or as part of the *Step* or *Solve* operations), then the mapped regions will be output for postprocessing.

Otherwise, the postprocessing results will be written after the mapping but before the first coupling step or iteration.

For more information, see:

Adding Instancing Objects

Defining Instancing for a Coupling Interface Side

Adding Instancing Objects

You can add instancing objects using either the GUI or the CLI. Using the steps indicated below, you may create multiple instancing objects as needed.

Note:

Instancing objects are not available if at least one side of an interface is an FMU participant.

When you add the first instancing object to the analysis, the `Library` singleton is added to the data model (if it does not already exist), with the new `Instancing` object defined beneath it.

For more detailed information on instancing object definition requirements, see `Instancing` in the *System Coupling Settings and Commands Reference* documentation.

Adding an Instancing Object in the GUI

1. Perform one of the following actions:

   - From the **Menu Bar**, select **Setup** → **Add Instancing**.
   - In the **Outline**, right-click the **Setup** branch and select **Add Instancing**.

   The **Instancing** branch is added to the **Library** branch, with the new instance object defined underneath it.

   Tip:

   Once the **Instancing** branch exists, you may add new instancing objects by right-clicking it and selecting **Add**.

   ---

2. Click the instancing object.

   Corresponding settings are shown in the **Properties** pane below.

3. From the **Rotation Axis** list, select one of the following options:

   - `ReferenceFrame` (default)

     Defines instancing based on an existing reference frame.

   - `PrincipalAxis`

Defines a rotation about one of the coordinate axes.

- `UserDefinedAxis`

  Allows for a rotation about an arbitrary axis defined by two coordinates.

4. Optionally, you may set the available parameters for the instancing object.

Adding an Instancing Object in the CLI

1. Add the instancing object by specifying an object name. (The example below changes the reference frame setting from its default value of ***`GlobalReferenceFrame`***.)

```
DatamodelRoot().Library.Instancing["Instance-1"].ReferenceFrame = "Frame-1"
```

   If they do not already exist, the `Library` singleton and the `Instancing` object container are added to the data model and the instancing object is created with the parameter value set as specified. Other instancing object parameters are set to their default values.

2. Optionally, you may set the following parameters for the instancing object:

   - **Rotation Axis** is set to `ReferenceFrame`

```
DatamodelRoot().Library.Instancing["Instance-1"].RotationAxis = "ReferenceFrame"
DatamodelRoot().Library.Instancing ["Instance-1"].InstancesInFullCircle = 8
DatamodelRoot().Library.Instancing ["Instance-1"].InstancesForMapping = 3
```

   - **Rotation Axis** is set to `PrincipalAxis`

```
DatamodelRoot().Library.Instancing["Instance-1"].InstancesInFullCircle = 8
DatamodelRoot().Library.Instancing["Instance-1"].InstancesForMapping = 3
DatamodelRoot().Library.Instancing["Instance-1"].RotationAxis = "PrincipalAxis"
DatamodelRoot().Library.Instancing["Instance-1"].Axis = "Z"
```

   - **Rotation Axis** is set to `UserDefinedAxis`

```
DatamodelRoot().Library.Instancing["Instance-1"].InstancesInFullCircle = 8
DatamodelRoot().Library.Instancing["Instance-1"].InstancesForMapping = 3
DatamodelRoot().Library.Instancing["Instance-1"].RotationAxis = "UserDefinedAxis"
DatamodelRoot().Library.Instancing["Instance-1"].AxisFrom = [0, 0, 0]
DatamodelRoot().Library.Instancing["Instance-1"].AxisTo = [0, 0, "100 [cm]"]
DatamodelRoot().Library.Instancing["Instance-1"].RotationalOffset = "45 [deg]"
```

Tip:

To create an instancing object in the default state, you may use the `Set-State()` command and the intended object path, as follows:

```
```

```
SetState(ObjectPath='/SystemCoupling/Library/Instancing:Instance-1', State={})
```

Defining Instancing for a Coupling Interface Side

Each side of a coupling interface has an `Instancing` setting that allows you define an instancing object for that side.

If valid instancing objects have been defined for the analysis, you may define an object on an interface side as follows:

- **For the GUI**, select an instancing object from the `Instancing` setting's drop-down menu.
- **For the CLI**, define an instancing object as shown below:

```
DatamodelRoot().CouplingInterface['Interface-1'].Side['One'].Instancing = 'Instance-1'
```

If no value is provided, then this setting defaults to ***None*** (no instancing is defined).

Setting a Participant Connection Timeout Interval

System Coupling offers command-line options that allow you to set a timeout interval for participant connection, specifying how long System Coupling waits for participants to connect after connections were requested (that is, via a *Solve* operation or the `Initialize()` command).

To do so, start System Coupling from the command line (either the CLI or the GUI) and use the `-C` or `--connectiontimeout` argument to specify the maximum time interval System Coupling should wait for participants to connect before aborting the coupling run.

The timeout interval is defined in seconds and must be specified as a positive real value greater than zero seconds. The default value is ***`3600 [s]`*** — that is, a timeout interval of 1 hour is defined. The examples below illustrate how to set different timeout intervals.

Example 19: Setting a 30-second timeout interval when starting the CLI on Windows

```
> "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling" -C 30
```

Example 20: Setting a 10-second timeout interval when starting the GUI on Windows

```
> "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling" -G -C 10
```

Example 21: Setting a 120-second timeout interval when starting the CLI on Linux

```
```

```
$ "$AWP_ROOT251/SystemCoupling/bin/systemcoupling" --connectiontimeout 120
```

If any of the participants have not connected when the specified timeout interval is exceeded, then System Coupling generates an error message and aborts the coupling run.

For more detailed information, see General Command-Line Options in the *System Coupling Settings and Commands Reference* documentation.

Using Interactive Solve Commands

System Coupling provides a series of advanced commands that enable an interactive solve process (as opposed to solving straight through, as when a coupling run is initiated by the `Solve()` command). These commands can be issued both from the System Coupling CLI and the **System Coupling** GUI's **Command Console**.

In general, an interactive solve process is advisable if you require more control over your solution. Interactive solve commands provide the ability to pause the run at designated points, enabling you to review its progress and make changes as needed.

Available interactive commands are described briefly below. For more detailed information on a given command, see the corresponding entry in the Commands section of the *System Coupling Settings and Commands Reference* manual.

`PrintSetup()`

Prints the **Summary of Coupling Setup** to the System Coupling Transcript.

`Initialize()`

Prepares System Coupling for the run, makes connections to participants, starts participants, and prints participant build information to System Coupling's Transcript.

Disabled when a solution is already in progress. Note that this command will raise an exception if another instance of System Coupling is solving in the current working directory.

If this command is (re)issued after participants have been started, then the subsequent command's arguments are used. If there are changes to a participant's `.scp` file, then the new file replaces the previous one.

`Step()`

Available for step-based coupled analyses. (For more information on iteration-based vs. step-based steady analyses, see Steady Analysis Type (p. 37).)

Specifies the number of coupling steps to be run before pausing the coupled analysis. If necessary, starts participants. Default value is ***1***.

Disabled when a solution is already in progress. This command will raise an exception if another instance of System Coupling is solving in the current working directory.

When you resume the solution, either by reissuing this command or running the `Solve()` command, System Coupling restarts the analysis at the point it left off and continues the solution with the next step.

`CreateRestartPoint()`

Writes a restart point to a Restart file in an initialized analysis.

This command signals System Coupling and each participant that a restart point should be created before the next coupling step or coupling iteration begins. The requested restart point is created in addition to any restart points created either by System Coupling (via the `OutputControl` data model settings) or by participant configurations.

Note:

Some participants write their restart files only when the coupling run resumes, so their files are not available immediately after the `CreateRestartPoint()` command is issued.

Once a restart point is created, you can close the analysis and then resume working with it later, as described in Restarting a Coupled Analysis (p. 142).

`Shutdown()`

Shuts down a coupling analysis which has been initialized.

This command signals System Coupling and each participant to create a restart point at the end of the last completed coupling step or coupling iteration and then shut down the analysis as soon as possible. System Coupling then ends the coupling run, disconnecting from participants and printing timing information to the Transcript and Log file (p. 332).

When instances of System Coupling started in GUI Server mode are disconnected, the GUI Server file (p. 331) is removed from the working directory.

Simultaneous Execution of Participant Solutions

System Coupling allows for groups of independent participants to execute their solutions simultaneously. If enabled, groups of participants which have no direct or indirect dependence on one another within a coupling iteration are solved simultaneously. System Coupling groups only those participants for which an iteration of the defined data transfer will introduce no dependence on any of the other participants in the group.

System Coupling's participant partitioning algorithms allocate simultaneous solutions across available compute resources, with the application of the algorithms modified to minimize core usage conflicts. The availability of partitioning algorithms varies according to the platform being used.

For more information, see:

System Coupling's Participant Grouping Process

Enabling Simultaneous Participant Solutions

Execution of Simultaneous Participant Solutions

Simultaneous Participant Solution Examples

System Coupling's Participant Grouping Process

For all coupled analyses, System Coupling performs an initial ordering of participants, as described in Participant Solution Sequencing (p. 81). When simultaneous participant solutions are enabled, additional resequencing is performed — first, when System Coupling creates groups of independent participants, and then, when it consolidates those groups (where possible) by combining them.

For details on System Coupling's participant grouping process, see:

Creation of Participant Groups

Consolidation of Participant Groups

Creation of Participant Groups

Using the ordered participant list obtained via the process described in Participant Solution Sequencing (p. 81), System Coupling groups the ordered participants based on their interdependencies with one another, as follows:

1. System Coupling creates the group, as follows:

    a. In the modified ordered participant list, identify the participants that have no dependencies with other participants on the list.

    b. Create the participant group, using the same criteria as previously:

    - If independent participants have been identified, then combine those participants into a group.
    - If no independent participants have been identified, then create a group with the first participant on the list as its only member.

    c. Adds the new group to the ordered list of groups, determining its position by applying the participant-ordering process (described in Participant Solution Sequencing (p. 81)) to the groups.

    d. Removes the grouped participant(s) from the ordered list of individual participants.

2. System Coupling repeats step 1 until no ungrouped participants remain on the individual participant list.

Consolidation of Participant Groups

Once an ordered list of participant groups has been created (as described in Creation of Participant Groups (p. 161)),System Coupling further consolidates the list by combining groups together, where possible. It combines the ordered groups (essentially grouping the groups) according to their interdependencies with one another, as follows:

1. System Coupling creates the first consolidated group, as follows:

a. In the ordered group list, identify the groups that have dependencies only on groups that come before them on the list.

b. Combine the identified groups into a consolidated group.

c. Add the consolidated group to the ordered list, determining its position by applying the participant-ordering process (described in Participant Solution Sequencing (p. 81)) to the groups.

d. Remove the combined groups from the ordered list.

2. System Coupling repeats step 1 until all eligible groups on the ordered list have been combined.

Enabling Simultaneous Participant Solutions

For information on enabling simultaneous participant solutions in a given user interface, see:

Enable Simultaneous Participant Solutions in the GUI

Enable Simultaneous Participant Solutions in the CLI

Enable Simultaneous Participant Solutions in the GUI

To enable simultaneous participant solutions in the GUI:

1. In the **Outline** tree, select the **Analysis Control** branch.

   Analysis settings are shown in the **Properties** pane.

2. Set **Allow Simultaneous Update** to **True**.

3. Set **Simultaneous Participants** to **Independent** or **All**.

Simultaneous participant updates are enabled.

Enable Simultaneous Participant Solutions in the CLI

To enable simultaneous participant solutions in the CLI, change the `AnalysisControl.AllowSimultaneousUpdate` setting to ***`True`***, as shown below:

```
DatamodelRoot().AnalysisControl.AllowSimultaneousUpdate = True
```

Simultaneous participant updates are enabled.

Execution of Simultaneous Participant Solutions

Simultaneous solutions are executed as parallel runs. In all cases, the solutions are allocated across available compute resources by System Coupling's participant partitioning algorithms. When simultaneous solves are enabled, the behavior of System Coupling's partitioning algorithms is modified to mitigate core-usage conflicts.

When a non-default algorithm is used, the corresponding partitioning functionality is unchanged. Whether solving individual participants or groups of simultaneously solved participants, the **Distributed Allocate Cores**, **Distributed Allocate Machines**, and **Shared Allocate Cores** algorithms behave as described in Parallel Processing for Coupling Participants (p. 166).

When the default **Shared Allocate Machines** algorithm is used, however, System Coupling adjusts partitioning behavior to minimize situations where multiple participants are attempting to use the same core at the same time. When this default algorithm is used, partitioning behavior is as follows:

- Individual participants with sequential solutions are partitioned using the specified **Shared Allocate Machines** algorithm.
- Participants that are grouped for simultaneous solution are partitioned as follows:
  - **Windows**: Member participants' resource allocation fractions are re-normalized and the group is partitioned independently of the other participants using the **Shared Allocate Cores** algorithm.
  - **Linux**: Member participants' resource allocation fractions are re-normalized and the group is partitioned independently of the other participants using the **Distributed Allocate Cores** algorithm.

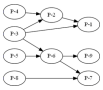
Simultaneous Participant Solution Examples

This section provides examples of coupling participant groupings for simultaneous solutions. Each example includes:

- Details of the ordering and grouping processes
- A diagram depicting dependencies and the final participant groupings

Example 22: Nine participants with multiple dependent participant groups

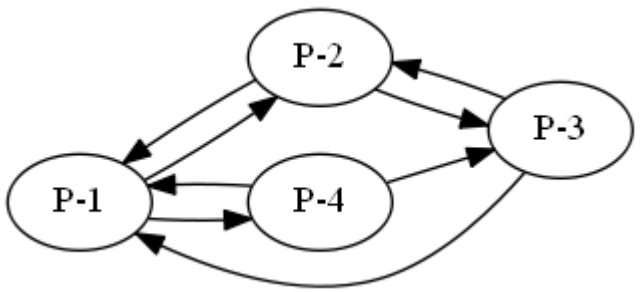
In this example, there are nine participants. None are sending Incremental Displacement.



Stage	Resulting Ordering/Grouping
Stage 1: Participant Ordering	[P-1, P-2, P-3, P-4, P-5, P-6, P-7, P-8, P-9]
Stage 2: Participant Grouping	[[P-3, P-4, P-5, P-8], [P-2, P-6], [P-1, P-7, P-9]]
Stage 3: Group Consolidation	[[P-3, P-4, P-5, P-8], [P-2, P-6], [P-1, P-7, P-9]]

Example 23: Four-participants with multiple dependent participant groups

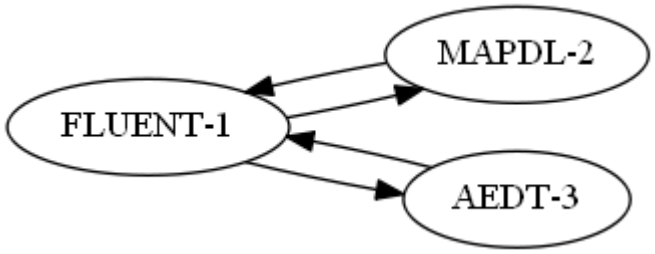
In this example, there are four participants. None are sending Incremental Displacement.



Stage	Resulting Ordering/Grouping
Stage 1: Participant Ordering	[P-1, P-2, P-3, P-4]
Stage 2: Participant Grouping	[[P-1], [P-2], [P-3], [P-4]]
Stage 3: Group Consolidation	[[P-1], [P-2, P-4], [P-3]]

Example 24: Coupled analysis with an MAPDL participant sending Incremental Displacement

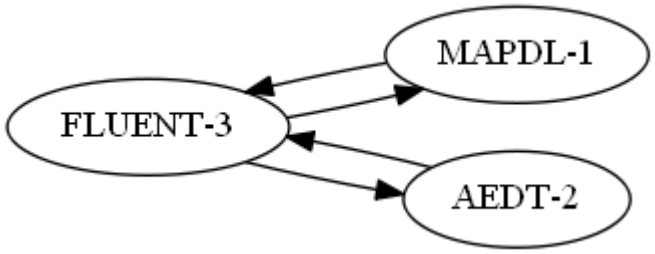
In this example, the MAPDL participant is sending Incremental Displacement to the Fluent participant.



Stage	Resulting Ordering/Grouping
Stage 1: Participant Ordering	[MAPDL-2, FLUENT-1, AEDT-3]
Stage 2: Participant Grouping	[[MAPDL-2], [FLUENT-1], [AEDT-3]]
Stage 3: Group Consolidation	[[MAPDL-2], [FLUENT-1], [AEDT-3]]

Example 25: Alternate scenario with an MAPDL participant sending Incremental Displacement

In this example, the configuration is the same as in the previous example, but the participant names are indexed differently.

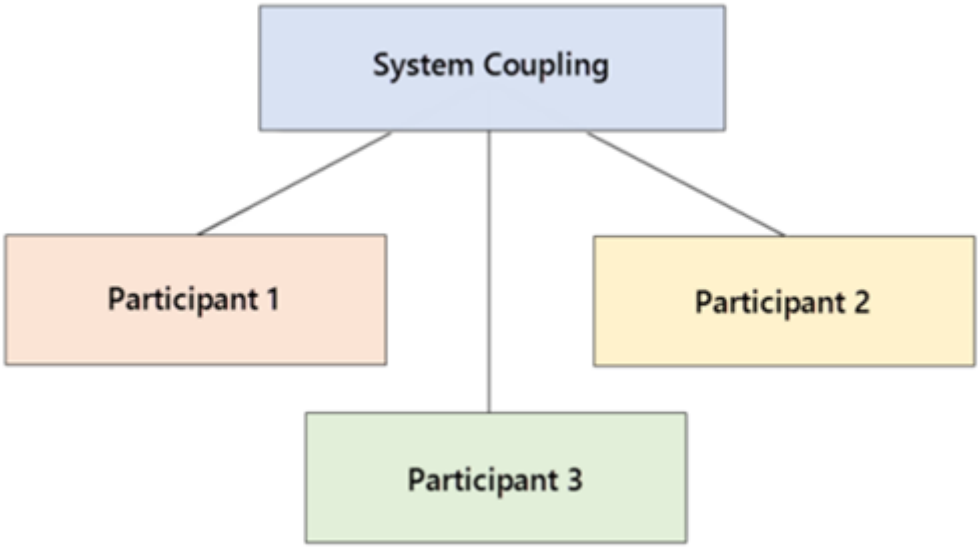


Stage	Resulting Ordering/Grouping
Stage 1: Participant Ordering	[MAPDL-1, AEDT-2, FLUENT-3]
Stage 2: Participant Grouping	[[MAPDL-1], [AEDT-2], [FLUENT-3]]
Stage 3: Group Consolidation	[[MAPDL-1, AEDT-2], [FLUENT-3]]

Using Parallel Processing Capabilities

System Coupling provides you with the ability to execute a coupled analysis using High-Performance Computing (HPC) resources in parallel. You can run the analysis on a single machine or distributed on two or more machines. Distributed runs are executed directly or indirectly with a queuing system, with Linux support for LSF, PBS Pro, UGE, and SLURM.

Figure 12: Communication between System Coupling and coupling participants



Note:

In the sections that follow, "machine" denotes a compute resource with multiple cores and a fixed amount of memory.

System Coupling always runs in parallel on two cores and has additional parallel execution features that work in conjunction with the automatic start-up of two or more participants. Load balancing of HPC resources is controlled via selecting the number of cores to be allocated to each participant and the algorithm for partitioning them. When participants are run in parallel along with System Coupling, it accelerates the heavyweight work performed during the simulation.

Note:

When running a large case via HPC, you can free up resources by disabling the creation of an initial snapshot. To do so, set the `OutputControl.WriteInitialSnapshot` to ***`False`***.

For more information, see:

Recommended Workflow for Parallel Execution

Parallel Processing for Coupling Participants

Parallel Options for System Coupling

Parallel Options for Participant Products

Job Scheduler Troubleshooting

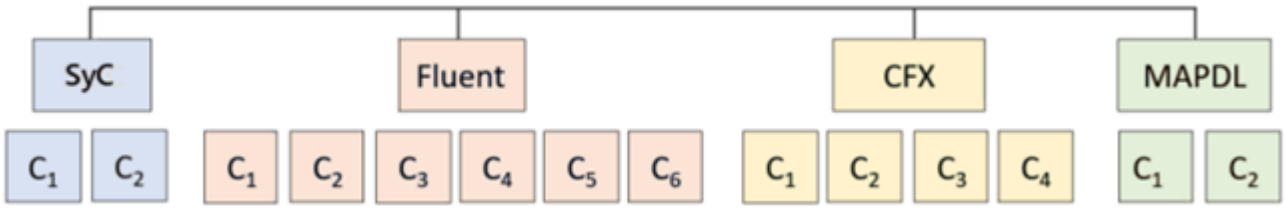
Recommended Workflow for Parallel Execution

1. Set up your analysis, as described in Preparing for a Coupled Analysis (p. 115).
2. To use parallel-specific command-line options when starting System Coupling, see Parallel Options for System Coupling (p. 184).
3. Set up your coupled analysis.
4. Edit your data model settings.
5. To run an individual participant product in parallel mode, specify the participant's command-line execution details. For more information, see Parallel Options for Participant Products (p. 184).
6. To run all coupling participants in parallel, specify how the participants are partitioned across compute resources. For more information, see Parallel Processing for Coupling Participants (p. 166).
7. Run the solution.

Parallel Processing for Coupling Participants

Parallel processing for co-simulation tends to be more involved than it is for single-solver simulations. A co-simulation involves, at minimum, three participants (that is, System Coupling and at least two coupling participants). Figure 13: Co-simulation with four coupling participants (p. 166) shows a co-simulation with four participants, each with different number of compute processes per participant. System Coupling provides participant partitioning capabilities, partitioning the simulation work across participants then allocating that work across available computing resources.

Figure 13: Co-simulation with four coupling participants



When setting up participants for parallel execution, you can set the following parameters to control the behavior of the run:

- **Machine List**: The list of machines to be used and, where applicable, the number of cores each has available for the parallel run. For more information, see Machine List and Core Counts (p. 167).
- **Participant Fractions**: The fractions specifying the core count or fraction of compute resources to be allocated to each participant. For more information, see Resource Allocation Fractions (p. 167).
- **Partitioning Algorithm**: The algorithm used to specify the resource allocation method (machines or cores) and parallel execution mode (local or distributed) to be used. For more information, see Partitioning Algorithms (p. 168).

The parallel execution details that were specified most recently — regardless of by what method — replace existing parallel values.

For more information, see:

Machine List and Core Counts

Resource Allocation Fractions

Partitioning Algorithms

Machine List and Core Counts

System Coupling allows you to specify the list of machines to be used and, where applicable, the number of cores available on each machine for a parallel run.

In many cases, you will not specify the machine list and core counts for a parallel run. For example, when the machine list and core count are handled by a load manager or job scheduler, you do not need to specify them.

However, if you do want to explicitly specify or review the machine list and available cores, you can use the command-line options described in the Coupling Participant Parallel Options section of the *System Coupling Settings and Commands Reference* manual.

When starting System Coupling (recommended):

Use the command-line options described in the Coupling Participant Parallel Options section of the *System Coupling Settings and Commands Reference* manual.

When running the System Coupling GUI:

Open the **Parallel solve setup** dialog (on the **Toolbar**, click ) and:

- For Linux, add details under **Machine Names and Core Counts**.
- For Windows, review auto-populated details under **Local Machine Core Counts**.

When running the System Coupling CLI or defining custom participant partitioning:

Run the `PartitionParticipants()` command, using the `MachineList` argument to specify machine name and core count details.

For more information, see Participant Partitioning Examples (p. 171).

Resource Allocation Fractions

When applying partitioning to coupling participants, you can use the `ParallelFraction` parameter to specify the core count or fraction of compute resources to be allocated to each participant. With regard to this setting, partitioning behavior depends on the parameter value and whether the default `SharedAllocateMachines` partitioning algorithm is used, as shown in the tables below.

In some cases, the resource allocation fractions provided by the default algorithm may not be optimal, depending on the participants involved and the co-simulation resource needs. For in-

stance, the Mechanical APDL structural solver might run more optimally on a lower core count than the Fluent solver, so you might want to set a lower fraction for MAPDL.

Note:

Resource allocation fractions are not supported for the ***Custom*** participant partitioning algorithm.

Table 23: Behavior when the ParallelFraction parameter is omitted and fractions are not provided

Partitioning algorithm	Behavior
`SharedAllocateMachines`	Each participant will run on all the allocated cores.

`SharedAllocateCores`
`DistributedAllocateMachines`
`DistributedAllocateCores`
	Each participant will run on an equal fraction of the total core count.

Table 24: Behavior when the ParallelFraction parameter is used and fractions are provided

Partitioning algorithm	Behavior
`SharedAllocateMachines`	If all the fractions are between ***0.0*** and ***1.0***, then each participant will run on exactly the specified fraction of the total core count. Otherwise, all fractions are normalized with respect to the largest fraction.

`SharedAllocateCores`
`DistributedAllocateMachines`
`DistributedAllocateCores`
	All fractions are normalized with respect to the sum of the fractions.

Partitioning Algorithms

System Coupling's algorithms provide for different permutations of parallel execution and resource allocation. By selecting an algorithm, you can control the parallel execution method (shared or distributed) and determine how participant work is allocated across compute resources (machines and cores).

When selecting a participant-partitioning algorithm, you specify the following information:

- **Resource allocation method**: Determines whether available compute resources will be allocated as machines or cores.

- **Parallel execution mode**: Determines whether participants will be run in shared parallel or distributed parallel (which are different than the local parallel and distributed parallel modes used for an individual participant). The table below compares the characteristics of each mode.

Table 25: Shared Parallel vs. Distributed Parallel execution modes

Characteristics	Shared Parallel	Distributed Parallel[1]
Resource sharing	Discrete participants share allocated machine resources.	Discrete participants are executed on non-overlapping machines so that they do not share machine resources.
Memory requirements	Ideally, the number of processes run on the machine should not exceed the total core count on a machine.	Useful when the co-simulation total memory requirements are greater than what is available for a shared simulation (that is, if the required memory to run in shared mode exceeds a machine's total memory).
Resource requirements	Multiple machines may be utilized, and the total number of required machines is dictated by the participant requiring the most resources.	Depending on how the cores are allocated, participants may end up sharing resources on some machines. The number of machines with shared resources may be as high as the total number of participants.
1: Distributed Parallel execution is supported only for Linux systems.		

For more information, see:

Choosing a Partitioning Algorithm

Partitioning Algorithm Pros and Cons

Participant Partitioning Examples

Applying Partitioning to Coupling Participants

Choosing a Partitioning Algorithm

When choosing a participant partitioning algorithm, consider the resources needed for each participant, the number of machines/cores that are available, and how those resources might be allocated to best meet your parallel-processing requirements.

The following algorithms are available:

- `SharedAllocateMachines (p. 173)`: Participants share both machines and cores. (default value)

> **Note:**
>
> When this algorithm is used and simultaneous solutions are enabled, partitioning behavior changes slightly. For more information, see Execution of Simultaneous Participant Solutions (p. 162).

- `SharedAllocateCores (p. 175)`: Participants share machines but not cores.
- `DistributedAllocateCores (p. 176)`: Participants minimally share cores and machines. (Linux only)
- `DistributedAllocateMachines (p. 177)`: Participants never share cores or machines. (Linux only)
- `Custom`: Participants use machines and cores as specified in the custom partitioning information you provide via the `PartitionParticipants()` command's `PartitioningInfo` argument.

For many co-simulations, the default ***`SharedAllocateMachines`*** algorithm is often the best choice. It behaves similarly to the methods used by individual solvers, providing for full utilization of all the allocated cores, and generally results in the best overall performance and scaling.

The other three "allocate" algorithms are provided as means of handling situations for which the default algorithm is not appropriate. For instance, if the total memory available on a machine is insufficient to handle running more than one participant, then you can use the non-default algorithm that best fits your parallel setup.

The ***`Custom`*** algorithm allows you to partition participants in a specialized way not offered by the other algorithms.

Partitioning Algorithm Pros and Cons

The tables below summarize the advantages and disadvantages generally associated with each of System Coupling's participant partitioning algorithms.

`SharedAllocateMachines`	
Pro:	Allows for full utilization of all cores (no idle cores)
Cons:	Machines must have enough memory for all participants (highest memory demand)

`SharedAllocateCores`	
Pro:	Cores are more evenly loaded – a better option for low memory machines
Cons:	Cores remain idle when individual participants are executing
	Participants must be tolerant of distributed memory execution
	The last participant may run on more machines if high core count machines are consumed by the other participants

`DistributedAllocateMachines` (Linux only)	
Pros:	Participants have complete access to all machine resources
	Useful if solving with participants that have high memory use
	Shared memory parallel gets higher utilization (better scaling for a participant)
Cons:	Cannot control the number of cores to be assigned to each participant
	Last participant may get fewer cores

`DistributedAllocateMachines` (Linux only)	
	Only useful if requesting that resources are allocated exclusively, so you know the number of cores on each machine

`DistributedAllocateCores` (Linux only)	
Pros:	Participants have complete access to all machine resources
	Useful if solving with participants that have high memory use
	Useful if you are requesting that resources are allocated exclusively
	Shared memory parallel gets higher utilization (better scaling for a participant)
Cons:	Cores remain idle when individual participants are executing
	You may get overlap (sharing) but the maximum is equal to the total participant count
	The last participant may run on more machines if high core count machines are consumed by the other participants

`Custom`	
Pros:	Useful if you need to partition participants in a way that is not possible using other algorithms
	Allows user to control the number of cores assigned to each participant
Cons:	Requires user intervention to manually query and assign machines and cores

Participant Partitioning Examples

This section provides a description and usage example for each of System Coupling's participant partitioning algorithms.

Shared Parallel: Allocate Machines

Shared Parallel: Allocate Cores

Distributed Parallel: Allocate Cores

Distributed Parallel: Allocate Machines

Custom Partitioning Examples

Note that all the examples share the following setup details:

- There are 57 cores with the same uneven distribution across nine machines, which are designated as `M0` through `M8`.
- System Coupling runs on the first machine on a single core.
- Each example assumes that the machine list has already been set up using one of the methods shown below.

The figures below show the three methods you can use to set up the machine list and core count for the examples that follow:

Example 26: Machine list and core counts specified using the --cnf command-line option (Linux only)

```
> "$AWP_ROOT251/SystemCoupling/bin/systemcoupling" --cnf="host0:12,host1:12,host2:12,host3:7,
host4:6,host5:3,host6:3,host7:1,host8:1"  -R run.py
```

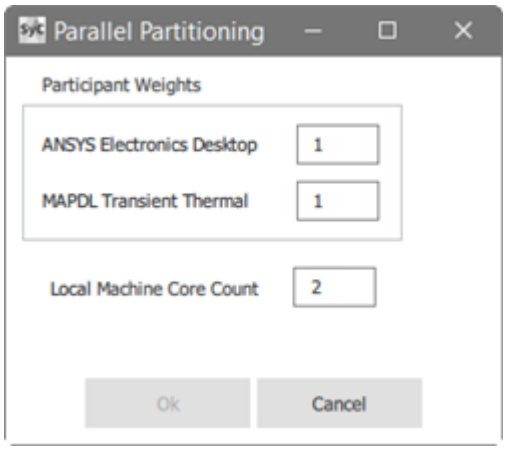
Example 27: Core counts specified using the -t command-line option

The specified core count cannot exceed the number of cores available on the local machine.

```
"%AWP_ROOT251%\SystemCoupling\bin\system coupling.bat" -t6  -R run.py
```

Example 28: Machine list and core counts specified using the System Coupling GUI (Linux)

Example 29: Core counts reviewed using the System Coupling GUI (Windows)



Example 30: Machine list and core counts specified using the PartitionParticipants() command (Linux)

```
> PartitionParticipants(MachineList = [
 {'machine-name' : 'host1', 'core-count' : 12},
 {'machine-name' : 'host2', 'core-count' : 12},
 {'machine-name' : 'host3', 'core-count' : 12},
 {'machine-name' : 'host4', 'core-count' : 7},
 {'machine-name' : 'host5', 'core-count' : 6},
 {'machine-name' : 'host6', 'core-count' : 3},
 {'machine-name' : 'host7', 'core-count' : 3},
 {'machine-name' : 'host8', 'core-count' : 1},
 {'machine-name' : 'host9', 'core-count' : 1}
 ])
```

Example 31: Machine list and core counts specified using the PartitionParticipants() command (Windows)

In this case, you must specify the name of the local machine, which can be found using the Python `socket` module, and then specify the core count.

```
import socket

    PartitionParticipants(MachineList = [{'machine-name' : socket.gethostname(),
    'core-count' : 12}])
```

Note:

Because all the partitioning examples that follow use the same machine/core setup, the machine list and core counts are the same for all of them. Only the partitioning algorithms and resource allocation fractions change for each example.

Shared Parallel: Allocate Machines

Participants share both machines and cores.

In the examples below, **Participant-1**, **Participant-2**, and **Participant-3** have been allotted fractions of 1, 1, and 1/5, respectively.

Example 32: GUI setup for the default Shared Allocate Machines algorithm with machines allocated by fractions (Linux only)

Note:

A Windows GUI example is not shown because distributed parallel is not supported on Windows. The dialog's machine name and core counts fields are auto-populated from the local host and cannot be edited.

Example 33: Data model setup for the SharedAllocateMachines algorithm with machines allocated by fractions

```
dm = DatamodelRoot()

dm.AnalysisControl.PartitioningAlgorithm = 'SharedAllocateMachines'
dm.CouplingParticipant['PARTICIPANT-1'].ExecutionControl.ParallelFraction = 1.0
dm.CouplingParticipant['PARTICIPANT-2'].ExecutionControl.ParallelFraction = 1.0
dm.CouplingParticipant['PARTICIPANT-3'].ExecutionControl.ParallelFraction = 1.0/5.0
```

Example 34: PartitionParticipants() setup for the SharedAllocateMachines algorithm with machines allocated by fractions

```
PartitionParticipants(AlgorithmName = 'SharedAllocateMachines',
        NamesAndFractions = [('PARTICIPANT-1', 1.0),
                             ('PARTICIPANT-2', 1.0),
                             ('PARTICIPANT-3', 1.0/5.0)])
```

Example 35: Resource allocation resulting from the SharedAllocateMachines examples

	M0 (12C)	M1 (12C)	M2 (12C)	M3 (7C)	M4 (6C)	M5 (3C)	M6 (3C)	M7 (1C)	M8 (1C)
System Coupling	1C	0C	0C	0C	0C	0C	0C	0C	0C
Participant 1	12C	12C	12C	7C	6C	3C	3C	1C	1C
Participant 2	12C	12C	12C	7C	6C	3C	3C	1C	1C
Participant 3	12C	0C	0C	0C	0C	0C	0C	0C	0C

Shared Parallel: Allocate Cores

Participants share machines but not cores.

The machines are sorted by core count and allocated in blocks, which are sized by the number of participants. This helps to ensure that each participant gets cores on machines with highest total core counts, keeping any inter-process communication local to a machine.

In the examples below, **Participant-1**, **Participant-2**, and **Participant-3** have each been allotted 19 cores.

Example 36: Data model setup for the SharedAllocateCores algorithm with even allocation of cores

```
DatamodelRoot().AnalysisControl.PartitioningAlgorithm = 'SharedAllocateCores'
```

Example 37: PartitionParticipants() setup for the SharedAllocateCores algorithm with even allocation of cores

```
PartitionParticipants(AlgorithmName = 'SharedAllocateCores')
```

Figure 14: Resource allocation resulting from the SharedAllocateCores examples

	M0 (12C)	M1 (12C)	M2 (12C)	M3 (7C)	M4 (6C)	M5 (3C)	M6 (3C)	M7 (1C)	M8 (1C)
System Coupling	1C	0C	0C	0C	0C	0C	0C	0C	0C
Participant 1	4C	4C	4C	7C	0C	0C	0C	0C	0C
Participant 2	4C	4C	4C	0C	4C	3C	0C	0C	0C
Participant 3	4C	4C	4C	0C	2C	0C	3C	1C	1C

Distributed Parallel: Allocate Cores

Participants minimally share cores and machines. Available for Linux.

The machines are sorted by core count and allocated into blocks, which are sized by the number of participants. This helps to ensure that each participant gets cores on machines with highest total core counts, keeping any inter-process communication local to a machine.

In the examples below, **Participant-1**, **Participant-2**, and **Participant-3** have each been allotted 19 cores.

Example 38: Data model setup for the DistributedAllocateCores algorithm with even allocation of cores

```
DatamodelRoot().AnalysisControl.PartitioningAlgorithm = 'DistributedAllocateCores'
```

Example 39: PartitionParticipants() setup for the DistributedAllocateCores algorithm with even allocation of cores

```
PartitionParticipants(AlgorithmName = 'DistributedAllocateCores')
```

Example 40: Resource allocation resulting from the DistributedAllocateCores examples

	M0 (12C)	M1 (12C)	M2 (12C)	M3 (7C)	M4 (6C)	M5 (3C)	M6 (3C)	M7 (1C)	M8 (1C)
System Coupling	1C	0C	0C	0C	0C	0C	0C	0C	0C
Participant 1	12C	0C	0C	7C	0C	0C	0C	0C	0C
Participant 2	0C	12C	0C	0C	6C	0C	1C	0C	0C
Participant 3	0C	0C	12C	0C	0C	3C	2C	1C	1C

Distributed Parallel: Allocate Machines

Participants never share cores or machines. Available for Linux.

The machines are sorted by core count and allocated in blocks, which are sized by the number of participants. This helps ensure that each participant gets cores on machines with highest total core counts, keeping any inter-process communication local to a machine.

In the examples below, **Participant-1**, **Participant-2**, and **Participant-3** have each been allotted 3 machines.

Example 41: Data model setup for the DistributedAllocateMachines algorithm with even allocation of cores

```
DatamodelRoot().AnalysisControl.PartitioningAlgorithm = 'DistributedAllocateMachines'
```

Example 42: PartitionParticipants() setup for the DistributedAllocateMachines algorithm with even allocation of cores

```
PartitionParticipants(AlgorithmName = 'DistributedAllocateMachines')
```

Example 43: Resource allocation resulting from the DistributedAllocateMachines examples

	M0 (12C)	M1 (12C)	M2 (12C)	M3 (7C)	M4 (6C)	M5 (3C)	M6 (3C)	M7 (1C)	M8 (1C)
System Coupling	1C	0C	0C	0C	0C	0C	0C	0C	0C
Participant 1	12C	0C	0C	7C	0C	0C	3C	0C	0C
Participant 2	0C	12C	0C	0C	6C	0C	0C	1C	0C
Participant 3	0C	0C	12C	0C	0C	3C	0C	0C	1C

Custom Partitioning Examples

Based on how available machines are assigned, the ***`Custom`*** algorithm allows you to configure machines to run either in shared parallel or distributed parallel, even within the same coupled analysis. It also allows for participant partitioning to be applied selectively, so that some participants run as specified by the algorithm while others run in their default mode.

In each of the examples that follow, it is assumed that each machine list includes multiple machines.

Example 44: Editing the data model to configure distributed participant execution

In this example, there are two participants and two machine lists.

Both participants are running in distributed parallel. Each participant is distributed across the machines in the list assigned to it.

```
machineList1 = [
    {'machine-name' : 'machine-1', 'core-count' : 5},
    {'machine-name' : 'machine-2', 'core-count' : 7}]
machineList2 = [
    {'machine-name' : 'machine-1', 'core-count' : 3},
    {'machine-name' : 'machine-3', 'core-count' : 7}]
partitioningInfo = {'MAPDL' : machineList1, 'CFX-2' : machineList2}

PartitionParticipants(PartitioningInfo = partitioningInfo)
```

Example 45: Editing the data model to configure distributed, shared, and serial participant execution

In this example, there are five participants and two machine lists.

The MAPDL participant runs distributed parallel across the machines in `machineList1`.

The CFX participants run in shared parallel across the machines in `machineList2`.

The remaining two participants are not included in the partitioning information. Partitioning is not applied, and participants run using their default settings.

```
partitioningInfo = {
    'MAPDL-1' : 'machineList1',
    'CFX-2' : 'machineList2',
    'CFX-3' : 'machineList2'}
```

Applying Partitioning to Coupling Participants

Partitioning coupling participants allows you to control how they are distributed across compute resources, specifying the partitioning algorithm to be applied, how available resources will be allocated, and what machines will be available for the parallel execution.

To apply partitioning to coupling participants, your coupled analysis must meet the following prerequisites:

- System Coupling has been started with one or more cores.
- At least one participant is defined.
- No participants are already started.

Note:

When simultaneous execution of participant solutions is enabled, the explicit application of partitioning is changed as described in Partitioning Algorithms (p. 168).

For more information, see:

Applying Participant Partitioning in the GUI

You may apply partitioning using the **System Coupling** GUI.

Note:

If you plan to use the ***Custom*** algorithm, use the steps outlined in Applying Custom Participant Partitioning (p. 182).

---

1. Specify the partitioning algorithm.

   a. In the **Outline** tree under **Setup**, select **Analysis Control**.

      Related settings are shown under **Properties**.

   b. Set **Partitioning Algorithm** to the algorithm to be applied to the analysis.

   For information on available algorithms, see Partitioning Algorithms (p. 168).

2. Specify the resource allocation fractions for each participant.

   a. Select one of the following options:

      - In the **Outline** tree under **Setup**, right-click **Coupling Participant** and select **Parallel solve setup**.
      - From the **Menu Bar**, select **Setup** → **Parallel Solve Setup**.
      - On the **Toolbar**, click  .

      The **Parallel Partitioning** dialog opens.

   b. Under **Participant Weights**, specify the resource weighting for each coupling participant as a real number.

   For more information, see Resource Allocation Fractions (p. 167).

3. Specify and/or verify the available machines and cores.

   The method by which machines and core counts are specified is determined by the platform being used, as follows:

   - **Windows**:

– The fields under **Local Machine Core Count** are auto-populated with the name of the local machine (**localhost**) and the number of cores available on that machine.

> **Note:**
>
> Because distributed parallel processing is not supported for Windows, specification of a machine and cores is not available here. However, you can use the `-t` command-line argument when starting System Coupling to specify core counts.

- **Linux**:
  – Under **Machine Names and Core Counts**, specify the name of each machine and the number of cores available on that machine.
  – Click **Add Machine** to add machines and their core counts to the list.

For details on this and other methods of setting this information, see Machine List and Core Counts (p. 167).

4. Close the dialog, as follows:
   - For Windows, click **Cancel**.
   - For Linux, click **OK**.

Applying Participant Partitioning in the CLI

When using System Coupling's CLI, apply partitioning to coupling participants using a combination of command-line arguments and data model edits.

> **Note:**
>
> If you plan to use the ***`Custom`*** algorithm, use the steps outlined in Applying Custom Participant Partitioning (p. 182).

1. Specify the number of cores to be used by the local machine.

   When starting System Coupling, use the `-t` command-line argument to specify the number of cores to be used when running on the local machine.

```
> "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling" -t4 -R run.py
```

> **Tip:**
>
> On a Linux system, you can explicitly set the machine list and the number of cores to be used by each machine. To do so, issue the `--cnf` command-line

argument when starting System Coupling. However, if a Linux job is being submitted to a job scheduler (that is, PBS, LSF, SLURM or UGE), then the machine list is automatically detected from the scheduler environment if you do not provide it via the command-line arguments (for example, `-t`, `--cnf`) or the `PartitionParticipants()` command.

For more information, see Machine List and Core Counts (p. 167).

For information on the `-t` and `--cnf` arguments, see Coupling Participant Parallel Options in the *System Coupling Settings and Commands Reference* manual.

2. Specify the partitioning algorithm.

   To specify the partitioning algorithm to be applied to the coupled analysis, edit the `AnalysisControl.PartitioningAlgorithm` setting in the data model.

   ```
   DatamodelRoot().AnalysisControl.PartitioningAlgorithm = 'DistributedAllocateCores'
   ```

   For information on available algorithms, see Partitioning Algorithms (p. 168).

   Note:

   - Distributed parallel processing is not supported for Windows systems.
   - When running either shared or distributed execution modes, CFX and Fluent participants should be the last participants added to the analysis because they tolerate distributed parallel processing well.

3. Specify the resource allocation fractions for each participant.

   To specify the core count or fraction of compute resources to be allocated to each participant, edit each participant's `ExecutionControl.ParallelFraction` setting in the data model.

   ```
   DatamodelRoot().CouplingParticipant['FLUENT-2'].ExecutionControl.ParallelFraction=1.0/5.0
   ```

   For more information, see Resource Allocation Fractions (p. 167).

Note:

The `PartitionParticipants()` command is available as an alternative method of applying partitioning to coupling participants.

Applying Custom Participant Partitioning

If you need to partition participants in a specialized way that is not offered by System Coupling's system-defined partitioning algorithms, you can use the ***`Custom`*** partitioning algorithm, which allows you to assign machines to participants as needed.

Note:

Resource allocation fractions are not used for custom participant partitioning.

To use custom partitioning, review the following sections:

Partitioning Information

Getting Lists of Available Participants and Machines

Applying Custom Partitioning by Running a Command

Applying Custom Partitioning in the GUI

Applying Custom Partitioning by Running a Script

Partitioning Information

When you run the `PartitionParticipants()` command, the `PartitioningInfo` argument accepts a dictionary specifying what machine resources are assigned to each participant. Each tuple pair in the dictionary consists of:

- a participant name as key
- a machine list (a list of dictionaries specifying available machines and their core counts) as value

When you set partitioning details using the `PartitioningInfo` argument, System Coupling uses the *`Custom`* partitioning algorithm automatically and blocks all other partitioning algorithms.

Note:

This means that if you attempt to use any other `PartitionParticipants()` argument in conjunction with the `PartitioningInfo` argument, you will receive an error.

If you provide incomplete information for the `PartitioningInfo` argument — that is, when either the name of an existing participant is not included or an included participant is associated with an empty machine list — then the affected participant executes using its default parallel settings.

Getting Lists of Available Participants and Machines

To get values for the `PartitioningInfo` argument, obtain lists of available participants and machines.

To get participants, query the `CouplingParticipant` object.

```
>>> DatamodelRoot().CouplingParticipant.GetChildNames()

['MAPDL-1', 'FLUENT-2', 'FLLUENT-3']
```

To get machines, run the `GetMachines()` command.

```
>>> GetMachines()

[{'machine-name' : 'machine-1', 'core-count' : 4}, {'machine-name' : 'machine-2',
'core-count' : 8}, {'machine-name' : 'machine-3', 'core-count' : 12}]
```

You may either obtain these lists as shown above or include the queries in a Python script, as shown in Applying Custom Partitioning by Running a Script (p. 184).

Applying Custom Partitioning by Running a Command

To apply custom partitioning, run the `PartitionParticipants()` command, using the `PartitioningInfo` argument to specify participants and the machines assigned to them.

Note:

- When you specify partitioning details using the `PartitioningInfo` argument, the ***`Custom`*** algorithm is used automatically.
- If `AlgorithmName` has been set to ***`Custom`*** manually (not recommended), then you must still set using the `PartitionParticipants()` command's `PartitioningInfo` argument.

In the example below, each of the three participants has one machine assigned to it. In this case, the `AlgorithmName` argument is not required because partitioning details are set by the `PartitioningInfo` argument.

```
>>>PartitionParticipants(
      PartitioningInfo = {
          'MAPDL-1': [{'machine-name' : 'machine-1', 'core-count' : 4}],
          'CFX-2': [{'machine-name' : 'machine-2', 'core-count' : 4}]
      }
    )
```

Applying Custom Partitioning in the GUI

To apply custom partitioning in the GUI, specify partitioning parameters by running the `PartitionParticipants()` command in the **Command Console**. For details, see Applying Custom Partitioning by Running a Command (p. 183).

When the command is executed, the **Algorithm Name** property is set to **Custom** automatically and other partitioning algorithms are blocked.

Applying Custom Partitioning by Running a Script

You can apply custom partitioning by running a script in System Coupling's CLI.

```
allMachines = GetMachines()

# all AEDT participants go on the first core
aedtMachines = []
aedtMachines.append({'machine-name' : allMachines[0]['machine-name'], 'core-count' : 1})

# all FLUENT participants go on the rest of the cores
fluentMachines = []
fluentMachines.append(allMachines[1])
fluentMachines.append({'machine-name' : allMachines[0]['machine-name'], 'core-count' :
    allMachines[0]['core-count'] - 1})

partitioningInfo = {}

 for participant in DatamodelRoot().CouplingParticipant.GetChildren():
    participantName = participant.GetName()
    if participant.ParticipantType == "AEDT":
        partitioningInfo[participantName] = aedtMachines
    else if participant.ParticipantType == "FLUENT":
        partitioningInfo[participantName] = fluentMachines

PartitionParticipants(PartitioningInfo = partitioningInfo)
```

Parallel Options for System Coupling

When running in parallel, System Coupling always runs on two cores. For information on parallel command-line options, see System Coupling Parallel Options in the *System Coupling Settings and Commands Reference* manual.

If you also plan to use parallel processing for the coupling participants (p. 166), then you may wish to review Coupling Participant Parallel Options.

Parallel Options for Participant Products

Coupling participant setup files do not include parallel processing information. To run an individual coupling participant product in parallel execution mode, you must introduce solver-specific parallel details using settings defined under the participant's `ExecutionControl` singleton.

When parallel customizations for an individual participant are needed, you may use these execution control settings instead of the partitioning functionality described in Parallel Processing for Coupling Participants (p. 166).

When System Coupling starts the coupled analysis execution, these arguments are picked up from the data model and used to start the participant solver. If the specified arguments conflict with default values set by System Coupling, the new values override the System Coupling defaults.

For more information, see:

Adding Solver-Specific Parallel Arguments

Using AEDT HPC Distribution Options

Adding Solver-Specific Parallel Arguments

All coupling participants have an `AdditionalArguments (p. 185)` execution control setting. You may use this setting to enter arguments defining parallel details for the participant.

Note:

Some participants populate this setting with default values. For details, see the participant's product documentation.

You may add solver-specific arguments either in the GUI or the CLI.

Adding solver-specific parallel arguments in the GUI

When using the **System Coupling** GUI, specify solver-specific parallel arguments by adding them to the **Additional Arguments** setting, as follows:

1. In the **Outline** tree under **Coupling Participant**, locate and expand the branch for the participant object.
2. Under the participant object, click **Execution Control**.

   The participant's execution controls are shown in the **Properties** pane.
3. Enter the solver-specific arguments into the **Additional Arguments** field.

Adding solver-specific parallel arguments in the CLI

When using the System Coupling CLI, specify solver-specific parallel arguments by adding them to the `AdditionalArguments` setting in the participant's execution controls, as shown in the example below:

Example 46: Adding CFX parallel arguments from the CLI on Windows

```
DatamodelRoot().CouplingParticipant['FLUENT-2'].ExecutionControl.AdditionalArguments
= cfx5solve -def model.def -par-dist 'hosta,hostb*2,hostc*4'
```

Using AEDT HPC Distribution Options

For AEDT coupling participants, additional parallel processing settings are defined under the participant's `ExecutionControl` singleton. Each setting corresponds to functionality available in Maxwell's **Analysis Configuration** dialog and as command-line arguments for HPC integration.

You can use these settings to configure Maxwell to use automatic or manual HPC distribution. For more information, see:

Using Automatic HPC Distribution Options

Using Manual HPC Distribution Options

Using Automatic HPC Distribution Options

System Coupling supports the use of ***auto-distribution***, a Maxwell solver setting that allows Maxwell to estimate optimal HPC usage for each Maxwell solver type. To use auto-distribution in a System Coupling analysis, set the `AutoDistributionSettings` execution control to ***`True`***.

Auto-distribution is an evolving capability that performs well in many circumstances, as follows:

- For Maxwell 2D Transient simulations, the automatic setting generally makes very effective usage of the **Time Decomposition Method** (**TDM**, or the ***`Transient Solver`*** HPC distribution type).
- For Maxwell 3D Transient simulations, the automatic setting attempts to use the **Time Decomposition Method** with an estimate and limits for memory usage. However, memory usage exceeding the system limits will result in a crash. In this case, you will need to use manual HPC distribution settings (see below).
- For Maxwell 3D Eddy Current simulations, the automatic setting uses shared-memory HPC by default. To use multiple nodes, you will need to manually set the distribution type to ***Solution Matrix***.

You may enable auto-distribution in System Coupling's GUI or CLI.

Enabling Auto-Distribution in the GUI

1. Select the AEDT participant's **Execution Control** branch.
2. In the **Properties** pane, set **Auto Distribution Settings** to ***`True`***.

Enabling Auto-Distribution in the CLI

Set `AutoDistributionSettings` to ***`True`***, ensuring that you use the internal participant object name assigned by System Coupling.

```
DatamodelRoot().CouplingParticipant['AEDT-1'].
ExecutionControl.AutoDistributionSettings = True
```

When auto-distribution is enabled, the task count is set to ***`-1`*** and the machine list is formatted as follows: `list=machineA:-1:<nCoresA>, machineB:-1:<nCoresB>,...`

Tip:

This setting corresponds to Maxwell **Use Automatic Settings** check box and `-auto` command-line argument (shown below).

```
-distributed -auto -machinelist list=hostA:-1:32,hostB:-1:24,hostC:-1:48
```

Using Manual HPC Distribution Options

To edit Maxwell's distribution settings manually, set System Coupling's `AutoDistribution-Settings` parameter to ***`False`***. This allows you to control the HPC distribution types and specify how many cores per task will be used to distribute the HPC solution.

Note:

Distributed parallel execution is not supported when auto-distribution is disabled. That is, when `AutoDistributionSettings` is set to ***`False`***, only one machine may be specified by the `-machinelist` argument.

When auto-distribution is disabled, the following parallel settings are available.

`IncludeHPCDistributionTypes`

Required. Enter a string list of the distribution type(s) to be included for a distributed run. Available distribution types are ***`Solution Matrix`*** and ***`Transient Solver`***. Empty by default.

Tip:

This setting corresponds to Maxwell's **Job Distribution** tab and `-includetypes` argument (shown below).

```
-distributed "include types=Solution Matrix, Transient Solver"
-machinelist list=hostA:32:32
```

`NumberOfCoresPerTask`

Enter an integer greater than or equal to one to indicate the number of parallel cores to be used per task. Default value is ***`1`***.

Any cores per task will be used as a shared-memory HPC speed-up (multi-threading) for each task. The number of tasks is determined as follows: `nTasks = floor(nCores / nCoresPerTask)`

By default, the number of tasks is equal to the number of cores. For example:

- If a machine has a core count of ***`32`*** and `NumberOfCoresPerTask` is set to ***`1`***, then there will be ***`32`*** tasks.
- If `NumberOfCoresPerTask` is changed to ***`2`***, then there are ***`16`*** tasks.

- If `NumberOfCoresPerTask` is changed to ***3***, then the floor function is applied and there are ***10*** tasks.

Tip:

This setting corresponds to the task count and core count for Maxwell's **Machines** tab and `-machinelist` argument (shown below).

```
-distributed "include types=Solution Matrix" -machinelist list=hostA:16:32
```

`BatchOptions`

Enter a string to specify the specified batch option arguments to be applied to the run.

Note:

When this option is used from the command line, the escape characters `\"` are required.

Tip:

This setting corresponds to Maxwell's **Options** tab and `-batchoptions` argument (shown below).

```
-distributed "include types=Solution Matrix" -machinelist list=hostA:32:32
-batchoptions "\"'Maxwell 3D/NumCoresPerDistributedTask'=10\""
```

Example 47: Setting distribution settings manually from the command line

When editing distribution settings from the command line, make sure to use the internal participant object name assigned by System Coupling.

```
DatamodelRoot().CouplingParticipant['AEDT-1'].ExecutionControl.AutoDistributionSettings = False
DatamodelRoot().CouplingParticipant['AEDT-1'].ExecutionControl.IncludeHPCDistributionTypes = ['Transient S
DatamodelRoot().CouplingParticipant['AEDT-1'].ExecutionControl.NumberOfCoresPerTask = 2
DatamodelRoot().CouplingParticipant['AEDT-1'].ExecutionControl.BatchOptions = "\"'Maxwell 3D/NumCoresPerDi
```

Job Scheduler Troubleshooting

To ensure that System Coupling works correctly with a queuing system job scheduler, see the following notes.

LSF

- Setting the LSF_ENABLED environment variable to **1** may be necessary to start Fluent and System Coupling correctly on some LSF systems.
- When using LSF job schedulers, the Mechanical APDL participant may not terminate cleanly at the end of the coupled analysis. System Coupling forcibly terminates all participants after one hour and prints a warning message in the transcript. The default timeout value (3600 seconds) can be adjusted using the `--disconnecttimeout` command line option (for example, `--disconnecttimeout 60`). Note that it is important to give the participant solver enough time to write the results files and perform any other cleanup operations, but otherwise this issue is benign and should not negatively impact results and/or the ability to perform restarts.

PBS

- System Coupling automatically adds the following environment variables, with the following values, when starting the Fluent participant:

  ```
  SCHEDULER_TIGHT_COUPLING=1

  PBS_ENABLED=1
  ```

  They control the following:

  – Scheduler tight coupling with different defaults for each scheduler type.

  – Dynamic process spawning a workflow to work under a single job allocation.

  – Some startup procedures related to the scheduler identification.

  This is necessary to start Fluent correctly on most PBS systems. Note that these environment variables are added only if they are not already defined. If you set these environment variables prior to starting System Coupling, those values are respected.

- To successfully execute a job on PBS, you must explicitly change the working directory to the directory in which the job was submitted by adding the following line to the job submission script before starting System Coupling:

  ```
  cd $PBS_O_WORKDIR
  ```

SGE

- Setting the SGE_ENABLED environment variable to **1** may be necessary to start Fluent and System Coupling correctly on some SGE systems.

SLURM

- The host names provided in the SLURM_JOB_NODELIST environment variable at runtime may not be valid for establishing an SSH connection. This is dependent on the system configuration. If this is causing issues with execution using SLURM, System Coupling can switch to using alternative host names. To enable the option to perform a test SSH connection and to use alternative host names if the SSH test fails, set the SYSC_SLURM_DO_SSH_TEST environment variable to **1** before starting System Coupling.

- System Coupling automatically adds the following environment variables, with the following values, when starting the Fluent participant:

  ```
  SCHEDULER_TIGHT_COUPLING=1
  ```

  ```
  SLURM_ENABLED=1
  ```

  They control the following:

  – Scheduler tight coupling with different defaults for each scheduler type.

  – Dynamic process spawning a workflow to work under a single job allocation.

  – Some startup procedures related to the scheduler identification.

  This is necessary to start Fluent correctly on most SLURM systems. Note that these environment variables are added only if they are not already defined. If you set these environment variables prior to starting System Coupling, those values are respected.

- For Mechanical APDL, the following command line argument is added:

  ```
  -mpiopt " "
  ```

  Passing this argument will cause Mechanical APDL to remove the `-rsh` SSH argument when starting compute nodes on Intel MPI. This is necessary to start Mechanical APDL correctly on most SLURM systems.

Running an Exported System Coupling Setup

You can set up a coupled analysis in Workbench (as described in Exporting a System Coupling Setup (p. 273)), export the setup, and load the coupled analysis into either of System Coupling's user interfaces for execution.

For instructions on running an exported coupling setup in one of System Coupling's user interfaces, see:

Running an Exported Partial Setup

Running an Exported Full Setup

Running an Exported Partial Setup

For information on running an exported partial coupling setup in System Coupling's GUI or CLI, see:

Running an Exported Partial Setup in the GUI

Running an Exported Partial Setup in the CLI

Running an Exported Partial Setup in the GUI

To run an exported partial coupling setup in the GUI, perform the following steps:

1. Start the **System Coupling** GUI in the coupled analysis working directory.

   a. From the Windows **Start** menu, select **Ansys 2025 R1** > **System Coupling 2025 R1**.

The **Select Folder** dialog opens.

b. Navigate to the newly created/specified coupled analysis working directory, select the directory, and click **Select Folder**.

The GUI opens in the working directory and the exported setup is loaded.

On the **Messages** tab, note that there are four "**Action Required**" icons, indicating that setup is still required for these items. Because the setup was only partially completed in Workbench, the data model is populated only with coupling participant information.

2. Specify **Solution Control** settings.

a. Click **Solution Control**.

Corresponding settings are shown in the **Properties** pane, with the **Duration Option** setting set to **End Time** by default.

b. Define the analysis duration according to the **DurationOption** value:

- If keeping the default **End Time** value, enter a value for **End Time**.
- If using **Number of Steps**, specify any missing setting values (these are highlighted in red).

c. Enter a value for **Time Step Size**.

3. Add a coupling interface, as described in Adding a Coupling Interface in the GUI (p. 125).

4. Add at least one data transfer, as described in Adding a Single Data Transfer in the GUI (p. 128).

On the **Messages** tab, note that all the issues requiring attention have been addressed.

5. Execute the analysis.

Right-click the **Solution** branch and select **Solve**.

Running an Exported Partial Setup in the CLI

To run an exported partial coupling setup in the CLI, perform the following steps:

1. Start the System Coupling CLI.

a. Open the shell for your operating system in the newly created/specified coupling working directory.

b. Execute the command for the platform you are using:

- **Windows**

```
> "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling"
```

- **Linux**

```
$ "$AWP_ROOT251/SystemCoupling/bin/systemcoupling"
```

The command prompt says, "`Starting Command Console...`", indicating that System Coupling has started.

2. Load the exported setup by running the `Open()` command without arguments.

```
Open()
```

The command prompt says, "`Reading settings`", indicating that the data model is being populated from the `Settings.h5` file. When you are returned to the prompt, this means that the setup is loaded.

3. Verify that the data model is fully populated and passes validation.

   a. Run the `PrintSetup()` command.

      The coupled analysis setup is written to the Transcript, confirming that the data model is populated.

   b. Scroll to the bottom of the Transcript output and verify that **Output Control** is the last section. (Any existing validation errors will be listed after this section.)

      Because the setup was only partially completed before being exported from Workbench, required values are missing. The resulting validation errors are written to the end of the output, as shown below:

```
+--------------------------------------------------------------------------+
| Errors were found during data model validation.                          |
+--------------------------------------------------------------------------+
| Error: TimeStepSize not defined for Transient analysis                   |
| Error: EndTime not defined for Transient analysis                        |
| Error: Create at least one interface for this coupled analysis.          |
+--------------------------------------------------------------------------+
```

4. Specify the missing `SolutionControl` settings.

   a. For ease of reference, create a variable for path of the `SolutionControl` object container.

```
sc = DatamodelRoot().SolutionControl
```

   b. Set the analysis end time.

```
sc.EndTime = '1.0 [s]'
```

c. Set the time step size.

```
sc.TimeStepSize = '0.1 [s]'
```

5. Add a coupling interface by running the `AddInterface()` command, as described in Adding a Coupling Interface in the CLI (p. 126).

```
AddInterface(SideOneParticipantName = 'AEDT-1',
      SideOneRegionNames = ['Volume5'],
      SideTwoParticipantName = 'MAPDL-2',
      SideTwoRegionNames = ['FVIN_1'])
```

6. Add a data transfer by running the `AddDataTransfer()` command, as described in Adding a Single Data Transfer in the CLI (p. 128).

```
AddDataTransfer(Interface = 'Interface-1',
     TargetSide = 'Two',
     SourceVariable = 'Loss',
     TargetVariable = 'HGEN')
```

7. Execute the analysis by running the `Solve()` command.

```
Solve()
```

Running an Exported Full Setup

For information on running an exported full coupling setup in System Coupling's GUI or CLI, see:

Running an Exported Full Setup in the GUI

Running an Exported Full Setup in the CLI

Running an Exported Full Setup in the GUI

To run an exported full coupling setup in the GUI, perform the following steps:

1. Start the **System Coupling** GUI in the coupled analysis working directory.

   a. From the Windows **Start** menu, select **Ansys 2025 R1** > **System Coupling 2025 R1**.

      The **Select Folder** dialog opens.

b. Navigate to the newly created/specified coupled analysis working directory, select the directory, and click **Select Folder**.

The GUI opens in the working directory and the exported setup is loaded. Because the setup was fully completed in Workbench, the data model is populated with all information needed to run the analysis.

2. Execute the analysis.

Right-click the **Solution** branch and select **Solve**.

Running an Exported Full Setup in the CLI

To run an exported full coupling setup in the CLI, perform the following steps:

1. Start the System Coupling CLI.

a. Open the shell for your operating system in the newly create/specified coupling working directory.

b. Execute the command for the platform you are using:

- **Windows**

```
> "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling"
```

- **Linux**

```
$ "$AWP_ROOT251/SystemCoupling/bin/systemcoupling"
```

The command prompt says, "`Starting Command Console...`", indicating that System Coupling has started.

2. Load the exported setup by running the `Open()` command without arguments.

```
Open()
```

The command prompt says, "`Reading settings`", indicating that the data model is being populated from the `Settings.h5` file. When you are returned to the prompt, this means that the setup is loaded.

3. Verify that the data model is fully populated and passes validation.

a. Run the `PrintSetup()` command.

The coupled analysis setup is written to the Transcript, confirming that the data model is populated.

b. Scroll to the bottom of the Transcript output and verify that **Output Control** is the last section. (Any existing validation errors will be listed after this section.)

Because the setup was fully completed before being exported from Workbench, values are specified for all required settings.

If there are no other validation errors, the coupled analysis is ready to solve.

4. Execute the analysis by running the `Solve()` command.

```
Solve()
```

Requesting Debugging and Diagnostic Output

System Coupling can generate diagnostics and debugging output to help you troubleshoot issues with coupled analyses. For more information, see:

Requesting Debug Log Files

Requesting Diagnostic Files

Requesting Debug Log Files

System Coupling can generate debug log files to record transactions between itself and its participant solvers. Log file content is controlled by the command-line options listed in Debug Logging Command-Line Options in the *System Coupling Settings and Commands Reference* documentation.

To request that debug log files are generated, perform the following steps:

1. Start System Coupling from the command line (in either the CLI or the GUI), using the `-l` or `--logLevel` option to specify a logging level of ***1*** or higher.

2. Optionally, you can use the following additional arguments to specify details for the log output:

   - Use the `--logDir` option to specify an output directory for the log files.

     Note:

     By default, the log files are written to System Coupling's working directory.

   - If the logging level is set to a value of ***2*** or higher, use the `--logMaxArrayElements` to specify the maximum number of data array elements that can be included in the file. (Default value is ***5***.)

When System Coupling is started, debug logging is activated.

When debug logging is activated, during the solution, System Coupling will write data to the following files, with "*date*" and "*time*" indicating when the run began:

SyC_Log_Controller_<*date*>_<*time*>.txt

Contains log information from the controller process.

SyC_Log_CNode<*N*>_<*date*>_<*time*>.txt

Contains log information from compute node "*N*"

Tip:

The log file for compute node `0` usually contains the most relevant information.

Example 48: When starting the CLI on Windows, request that Level 1 debug log files are written to the default directory

Log files containing RPC calls will be written to System Coupling's working directory.

```
> "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling" --logLevel=1
```

Example 49: When starting the GUI on Windows, request that Level 2 log files are written to the specified directory

Log files containing RPC calls and diagnostics will be written to the **SyC/DebugOutput** directory within System Coupling's working directory.

```
> "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling" -G --logLevel=2 --logDir=SyC/DebugOutput
```

Example 50: When starting the CLI on Linux, request that Level 3 log files are written to the default directory

Log files containing RPC calls/diagnostics and a maximum of three heavyweight data array elements will be written to System Coupling's working directory.

```
$ "$AWP_ROOT251/SystemCoupling/bin/systemcoupling" --l3 --logMaxArrayElements=3
```

Requesting Diagnostic Files

System Coupling can generate files containing diagnostic information for mapping weights, interpolation, interfaces, and data transfers. These capabilities are available as beta functionality. For more information, see the following sections in the *System Coupling Beta Features* documentation:

- Export Coupling Interface Diagnostics

- Generate Weights and Interpolation Diagnostic Files

Reviewing Coupled Analysis Output and Results

Once the solution has completed, you can review System Coupling's output and perform postprocessing on the coupled analysis results. You can use any of the following methods:

- Review System Coupling's Transcript/Log file and any supplemental diagnostic files you have requested in the `scLog.scl` file, the CLI console, or the GUI's **Command Console** and **SC Transcript** tabs
- Review coupling participant transcripts in the **System Coupling** GUI
- View solution convergence diagnostics output as charts in the **System Coupling** GUI or as `.csv` files
- Load individual participant results into their respective postprocessing applications
- Load System Coupling's results into Ansys EnSight for visualization, animation, and postprocessing

For more information, see:

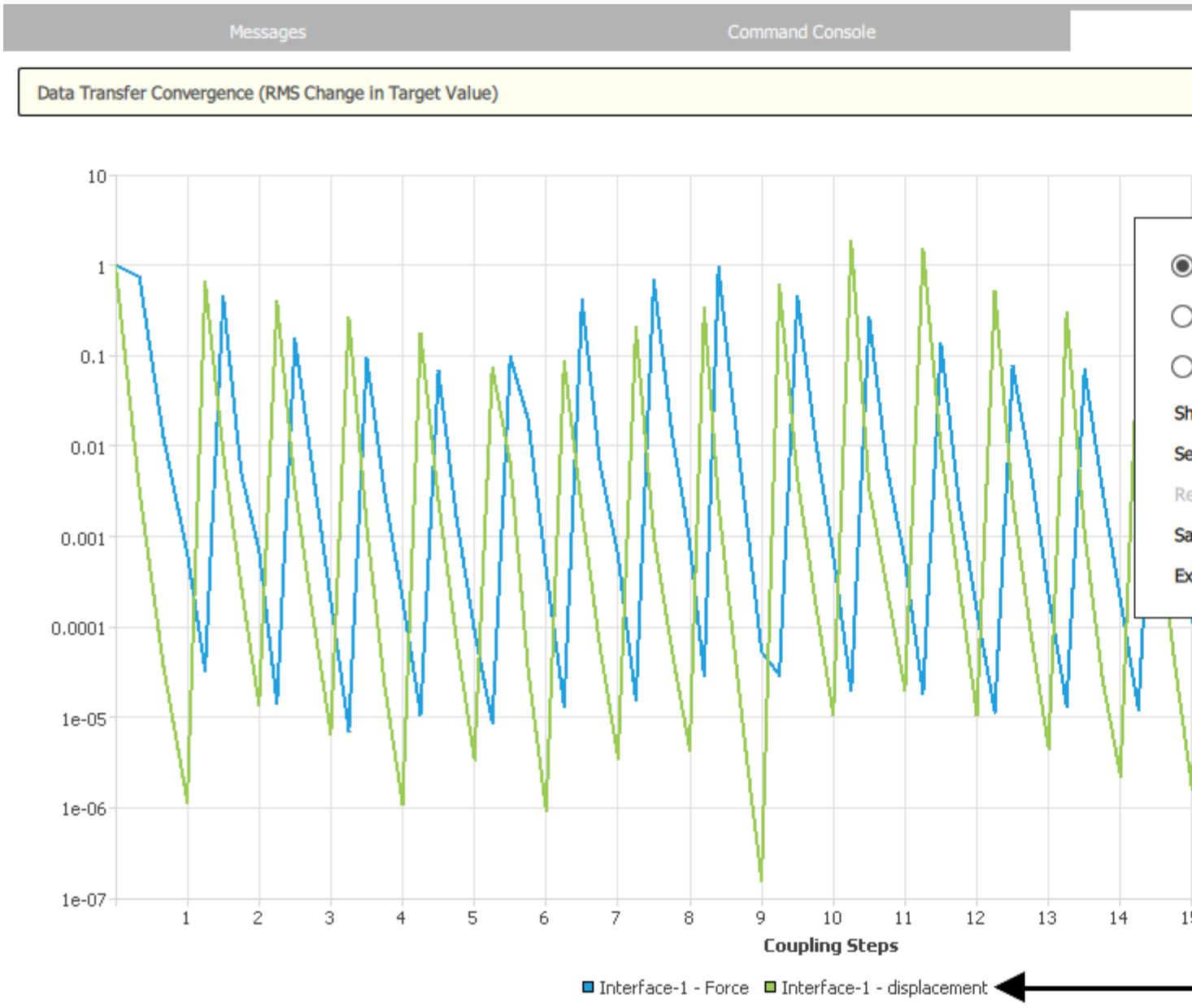
Reviewing Convergence Diagnostics Charting Output

Postprocessing Coupling Results in EnSight

Reviewing Convergence Diagnostics Charting Output

During the execution of a coupled analysis, System Coupling generates convergence diagnostic data and plots it on the **System Coupling** GUI's **Chart** tab. Charts are available in step, iteration, and simulation-time formats, with the step-based charts shown by default. Additionally, the convergence diagnostics data can be written to a `.csv` file for analyses run either in System Coupling's GUI or in its CLI.

Figure 15: Chart tab in the System Coupling GUI



Diagnostic data are updated dynamically in both the charted and `.csv` output formats, so both can be used to monitor convergence as the coupling run progresses and or to review the final set of convergence data after the run completes (or is stopped).

By right-clicking anywhere on the tab, you can access a context menu (p. 201) with options for performing various charting operations.

After a restart, System Coupling resumes updating the same plot lines on existing charts, unless significant changes have been made to the data model (for example, changes to interfaces or data transfers, changed setting values). When there are data model changes, System Coupling generates new charts, starting with the first iteration of the restarted run.

When System Coupling opens or connects to a coupled analysis:

- If the coupled analysis was accessed by switching to a different working directory, existing charting data from the previous working directory is removed.
- Charts resume at the current solution state and show the history of each data set.

For more information, see:

Core Charting Functionality

Chart Tab Context Menu

Viewing the Convergence Diagnostics Chart

Viewing Data Transfer Quantity Diagnostic Charts

Writing Charting Diagnostics to CSV Files

Core Charting Functionality

The **System Coupling** GUI's **Chart** tab is shown by default when a solution begins — regardless of whether it is the initial run, a restarted run, or a reconnected run — and relevant data becomes available.

Only one chart can be viewed at a time, and the Convergence Diagnostics (p. 202) chart is shown by default until you change the chart selection.

The following core functionality is available for all charts:

Active chart selection

The chart-selection drop-down menu at the top of the tab allows you to select a chart from among the charts generated for the simulation. The name selection also serves at the title for the active chart.

Step, Iteration, and Simulation Time chart views

All charts can be viewed in both a step-based and an iteration-based format. Transient charts may also be viewed in a simulation-time format. The step-based chart is shown by default, but you can switch between views using the chart view context menu (p. 201) options.

Steps, iterations, and simulation time are plotted dynamically along the X axis as charting data are generated, as follows:

- Labels are shown as simple powers of 10, with the number of labels adjusting according to the extent of the axis.
- Steps are plotted with equal spacing.
- Iterations are distributed equally within the parent step. With the addition of each new iteration, all existing iterations within the current step are redrawn.
- Simulation time is plotted at intervals consistent with whether the steps are of constant or non-constant duration. For the step that is in progress, the value of the most recently available iteration is shown as the current end time. For each completed step, the value for the final iteration of the step is shown as the step end time.

By default, charts show all steps/iterations or the full duration of the simulation. You can specify the number of steps/iterations or time interval to be shown using the visibility context menu (p. 201) options.

Data set plot lines

Relevant data are plotted on an auto-scaled XY line graph, with a separate line for each individual data set. A legend at the bottom of the tab indicates the name of each data set and the color of the corresponding plot line.

All data sets are selected (that is, all plot lines are shown) by default, but you can show or hide individual data sets using the **Show/hide data sets** context menu (p. 201) option.

X-axis chart controls

Zoom:

- Zoom controls allow you to zoom in on the X axis. Dynamic updates pause while the zoom function is active.
- To zoom in, click anywhere inside the tab, then either use the mouse wheel or hold down the **Ctrl** key and use the plus and minus keys (**+** and **–**) to adjust the zoom level.
- To reset the zoom level and resume chart updates, select the **Reset chart** option from the context menu (p. 201).

Pan:

- Pan controls are available when the zoom function is active and/or when you have specified the number of steps or iterations to be shown. Dynamic updates pause when the panning function is used.
- To pan, click anywhere inside the tab, then either hold down the left mouse key and drag the mouse or use the or the left/right arrow keys to move along the X axis.
- To reset the pan location and resume chart updates, select the **Reset chart** option from the context menu (p. 201).

Charting output

Coupling interface charting data can be output to `.csv` files, either automatically or on-demand. For more information, see Writing Charting Diagnostics to CSV Files (p. 206).

An image of the active chart can be exported as a `.png` file using the **Save chart image** context menu (p. 201) option.

Chart Tab Context Menu

By right-clicking anywhere on the **Chart** tab, you can access a context menu with options for performing various chart operations. The table below provides details on each available option.

Table 26: Chart context menu options

Option	Description
Step count view **Iteration view** **Simulation time view**	These options allow you to select a chart view based on steps, iteration, or simulation time. The step-based view is shown by default. Once specified, the chart view setting persists for the duration of the System Coupling session or until a different view is selected.
Show/hide data sets	Controls the visibility of the plot lines for individual data sets. By default, all data-set plot lines are shown. When selected, opens a dialog that allows you to select/deselect generated data sets data sets to show/hide their plot lines in the chart. Once specified, data set visibility settings persist for the duration of the System Coupling session or until a different option is selected.
Set step visibility **Set iteration visibility** **Set time interval to show**	Each option is available when the corresponding chart view is selected. Controls the number of steps, the number of iterations, or the time interval that is shown on the displayed charts. Selecting an option opens a dialog with corresponding options. For ***step-based*** or ***iteration-based*** charts, the following options are available: • **Show all steps/iterations**: Charts shows all the steps or iterations available for the simulation. (default) • **Specify steps/iterations to be shown**: Charts show the specified number of the most recent steps/iterations in the simulation. When selected, the following setting becomes available: – **Number of steps/iterations**: Accepts integers between **1** and the number of steps/iterations available for the simulation. For ***simulation time-based*** charts, the following options are available: • **Show from start of simulation**: Charts show the full duration of the simulation. (default)

Option	Description
	• **Specify time interval to be shown**: Charts show the specified time interval. When selected, the following setting becomes available: – **Time interval to be shown**: Accepts real numbers between **0** and the end time of the simulation, in seconds. Once specified, step/iteration/time visibility settings persist for the duration of the System Coupling session or until a different option is selected.
Reset chart	Available when zoom and/or pan functionality is active. Resets the chart back to its original zoom level and/or pan location and allows chart updates to resume.
Save chart image	Saves the active chart as a `.png` file. When selected, opens a **Save chart image** dialog that allows you to specify a location and a file name for the image.
Export CSV data for all interfaces	Exports charting convergence diagnostics for all coupling interfaces to `.csv` files. For more information, see Writing CSV Charting Output On Demand (p. 207).

Viewing the Convergence Diagnostics Chart

When a solution begins, System Coupling generates a single ***Convergence Diagnostics*** chart for the analysis and shows it by default on the **Chart** tab. This chart includes the convergence data for all data transfers across all interfaces in the analysis.

On this chart, System Coupling plots all the convergence diagnostics — that is, normalized differences per coupling iteration for coupling participants, coupling interfaces, and data transfers — on an XY graph with logarithmic scaling of the Y axis. Each diagnostic is a separate data set with its own line on the chart. At the end of each iteration, the updated value for each diagnostic is added as a new data point.

On the legend, the data sets are identified by interface and data transfer name. The axes of the graph are defined as follows:

- **Y Axis**: End-of-iteration values for each diagnostic.
- **X Axis**: Steps/cumulative iterations/time intervals for the simulation.

Figure 16: Step-based Convergence Diagnostics chart

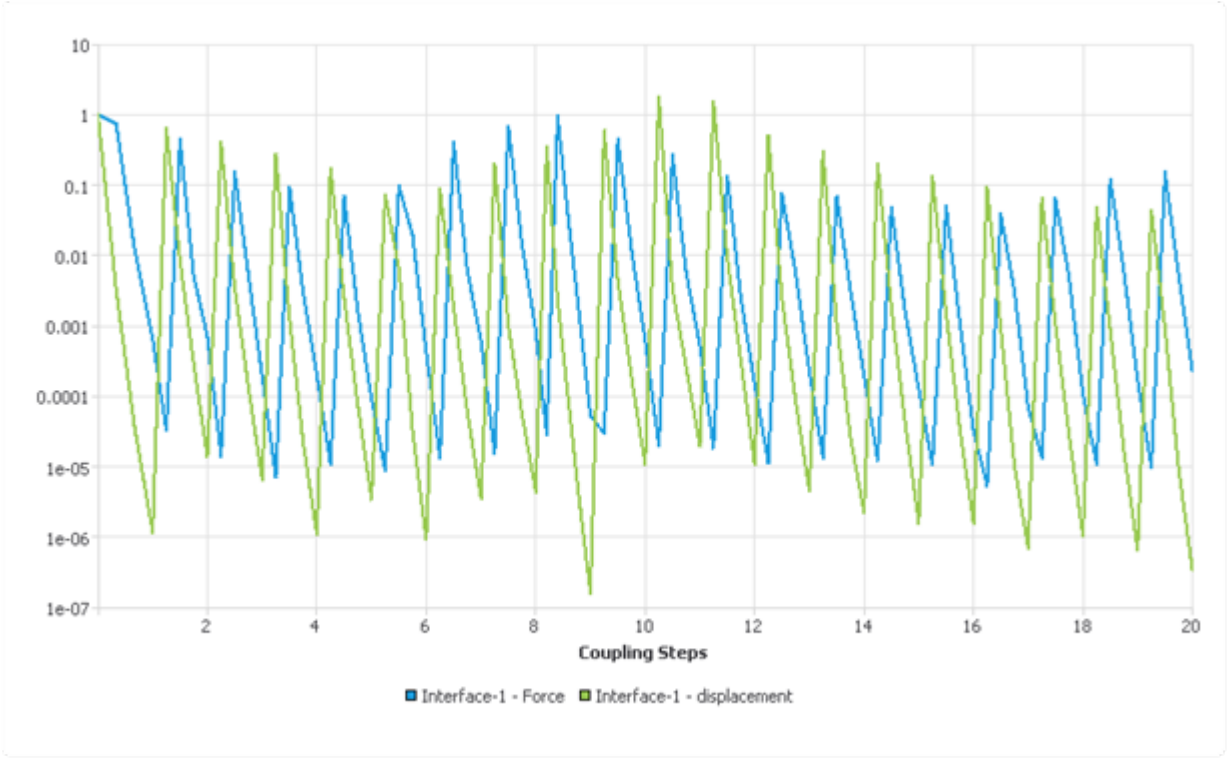


Figure 17: Iteration-based Convergence Diagnostics chart

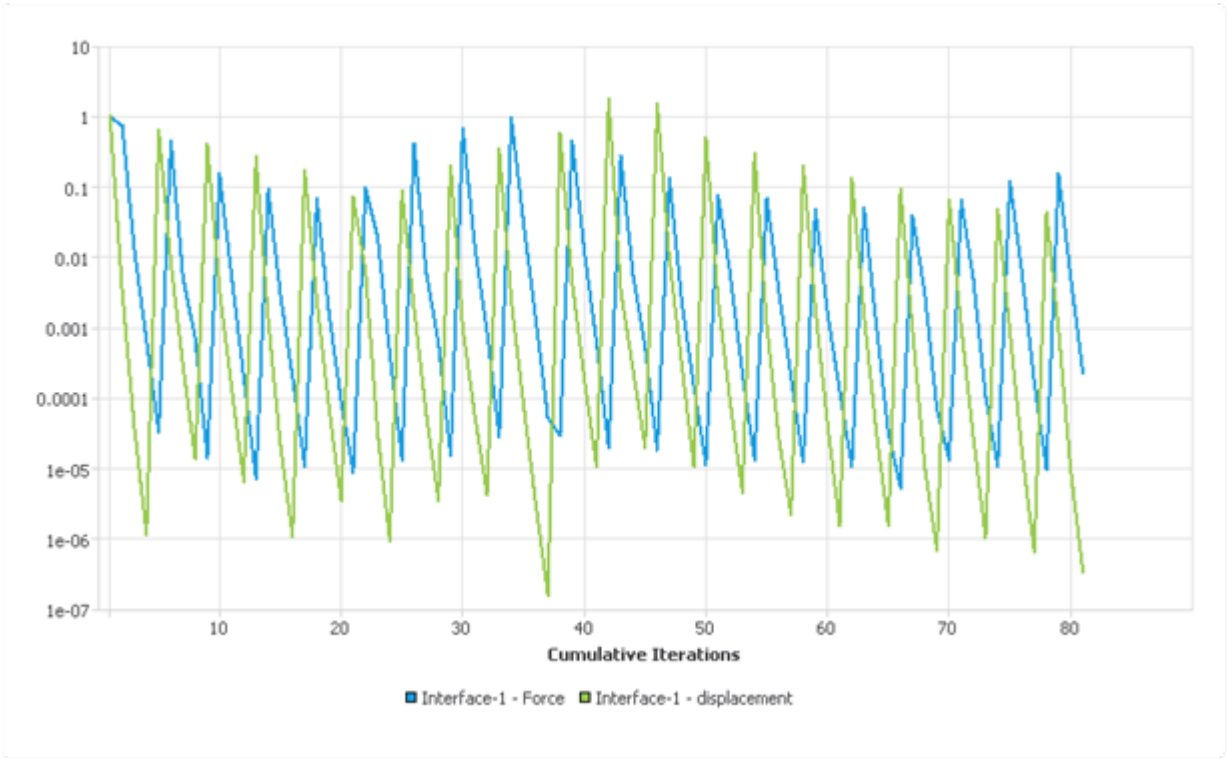
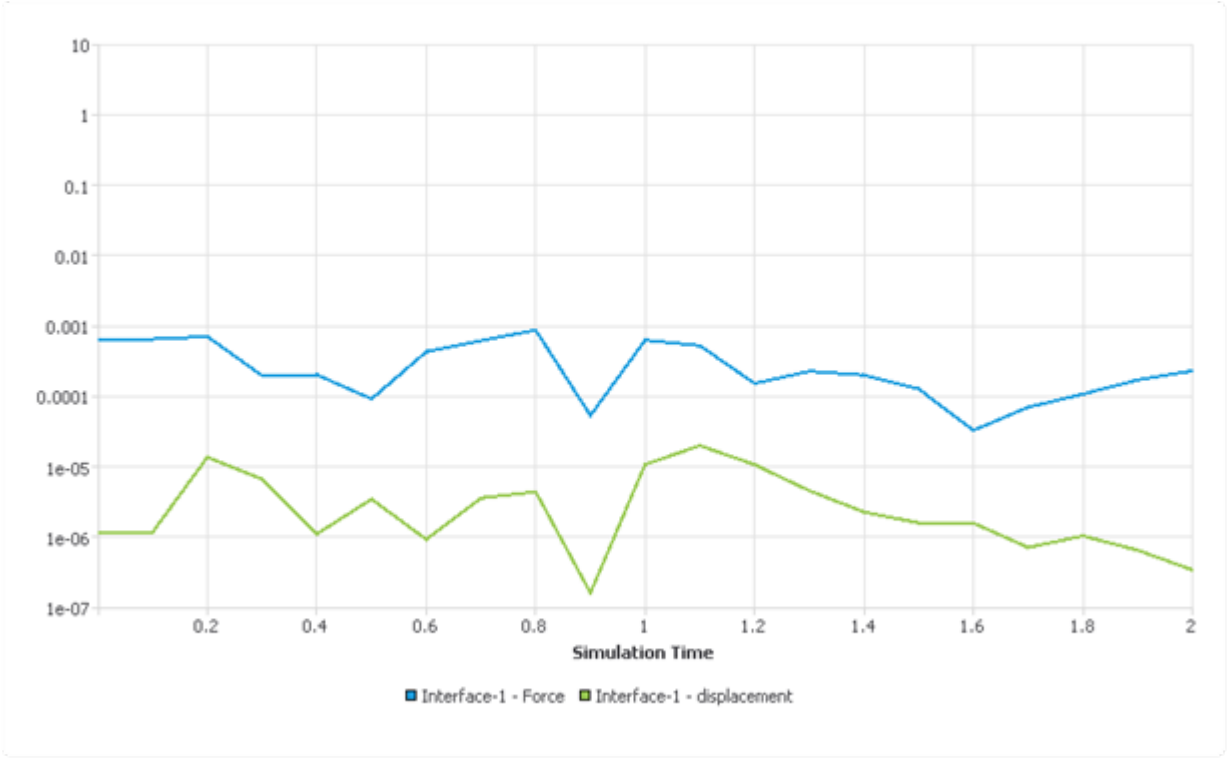


Figure 18: Simulation Time-based Convergence Diagnostics chart



Viewing Data Transfer Quantity Diagnostic Charts

System Coupling generates one or more ***Data Transfer Quantity Diagnostics*** charts for the analysis. One chart is generated for each data quantity type, and it includes data for all transfers of that quantity across all coupling interfaces in the analysis.

On this chart, System Coupling plots data transfer quantity diagnostics — that is, the **Sum** for transfers of extensive variables and an area or volume **Weighted Average** for transfers of intensive quantities — on an XY graph with linear scaling of the Y axis. Each transfer of the given quantity is a separate data set with its own line on the chart. At the end of each iteration, the updated diagnostic value is added as a new data point.

On the legend, each data set is identified by the name of the corresponding data transfer.

The axes of the graph are defined as follows:

- **Y Axis**: End-of-iteration diagnostic values for each data transfer.
- **X Axis**: Steps/cumulative iterations/time intervals for the simulation.

Figure 19: Step-based Data Transfer Quantity Diagnostic chart showing the Weighted Average

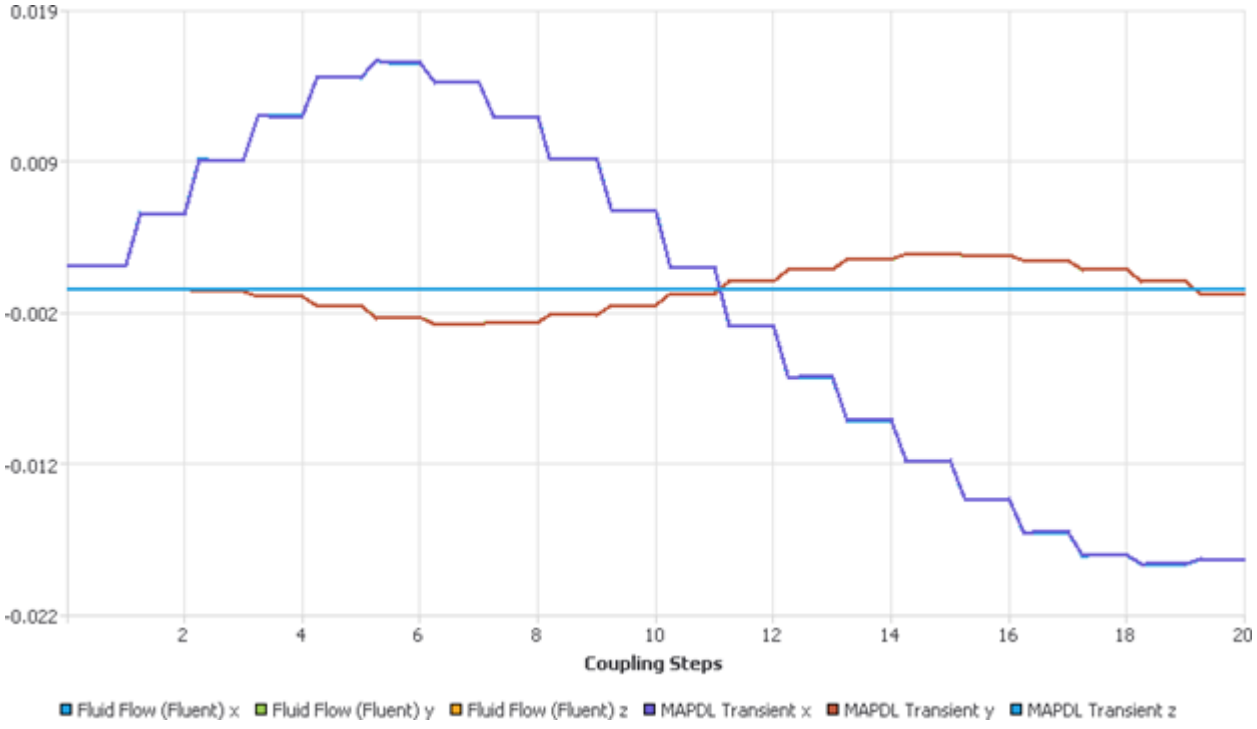


Figure 20: Iteration-based Data Transfer Quantity Diagnostic chart showing the Sum

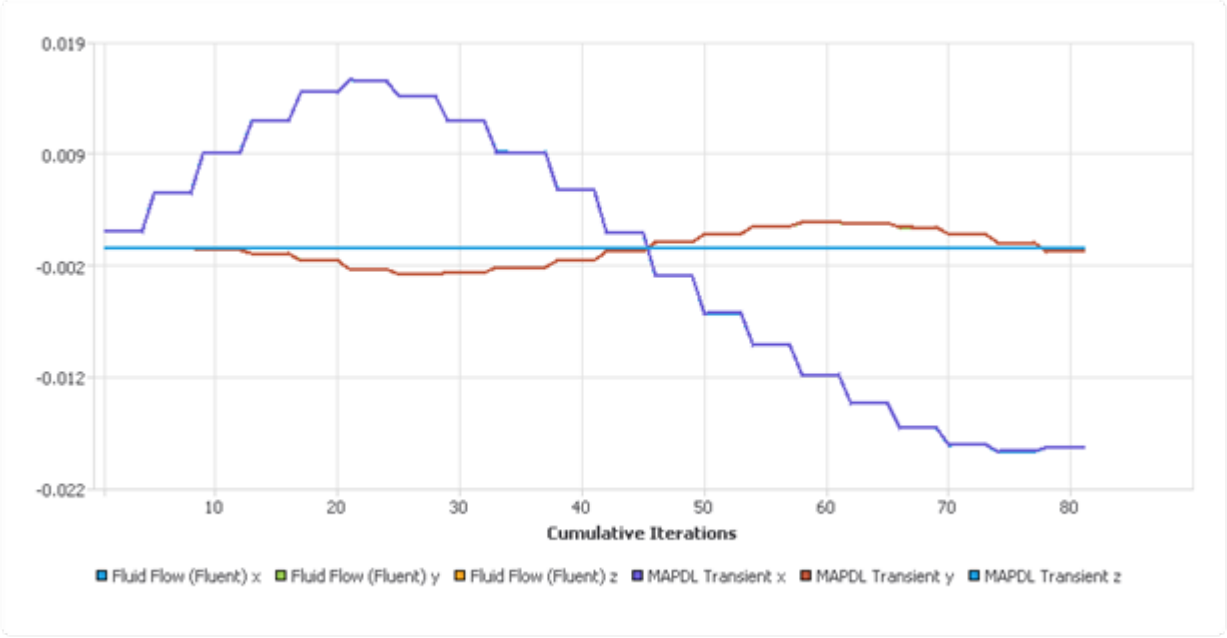
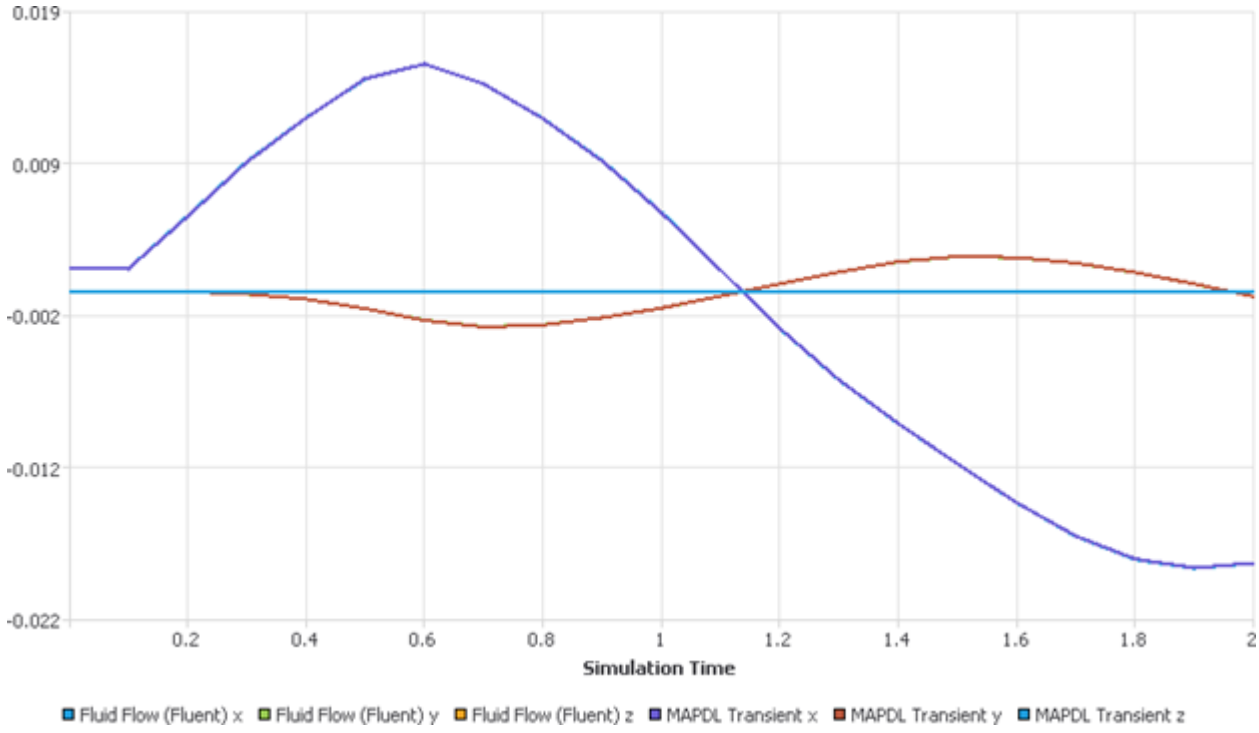


Figure 21: Simulation Time-based Data Transfer Quantity Diagnostic chart showing the Weighted Average



Writing Charting Diagnostics to CSV Files

System Coupling can write its charting convergence diagnostics to `.csv` files, both dynamically as the solution executes and on-demand for data that has already been generated.

For more information, see the following sections:

Generating Dynamic CSV Charting Output During the Solution

Writing CSV Charting Output On Demand

Generating Dynamic CSV Charting Output During the Solution

You can configure System Coupling to write convergence charting data to a `.csv` file as the analysis is executed. This capability is controlled by the **Output Control | Generate CSV Chart Output** data model setting.

To enable the generation of `.csv` output during execution, you can use either of the following methods:

Enabling Dynamic CSV Charting Output in the GUI

In the **Outline** tree under **Solution**, click **Output Control** and set the **Generate CSV Chart Output** setting to **True**.

Enabling Dynamic CSV Charting Output in the CLI

In the data model, set the `OutputControl.GenerateCSVChartOutput` setting to ***True***, as shown below:

```
>>> DatamodelRoot().OutputControl.GenerateCSVChartOutput = True
```

When the solution starts, System Coupling writes charting data to `.csv` files generated in its **SyC** coupling output directory. One file is created per interface, with each named according to the convention `<InterfaceName>.csv`, where "*InterfaceName*" is the internal name of the interface object in the data model (that is, not the display name shown in the GUI).

As the solution progresses, each file is updated with its interface's convergence and transfer data for each iteration.

Writing CSV Charting Output On Demand

If `.csv` charting output is enabled, as described in Generating Dynamic CSV Charting Output During the Solution (p. 206), you can request that all existing convergence charting data for a completed solution are written to a `.csv` file.

To request on-demand `.csv` charting output, you can use either of the following methods:

Requesting CSV Charting Output in the GUI

Right-click anywhere on the **Chart** tab and select the **Export CSV data for all interfaces** context option.

Requesting CSV Charting Output in the CLI

Run the `WriteCsvChartFiles()` command, as shown below:

```
>> WriteCsvChartFiles()
```

When the operation is executed, existing charting data are written to `.csv` files in the **SyC** folder of System Coupling's working directory. One file is created per interface, with each named according to the convention `<InterfaceName>.csv`, where "*InterfaceName*" is the internal name of the interface object in the data model (that is, not the display name shown in the GUI).

Note:

Files generated will overwrite any `.csv` charting files that already exist, including any that were written during the execution of the solution.

Postprocessing Coupling Results in EnSight

System Coupling generates coupling output in EnSight file formats, which makes it possible to load results into for high-performance 3D postprocessing. By default, System Coupling automatically produces postprocessing files at initialization, after each restart point, and at the end of the run. These

files can be loaded into EnSight for interactive postprocessing, including visualization and animation, of coupled analysis results.

Note:

EnSight output will not be created if all interfaces in the analysis contain sides without regions defined. However, results are written for the interface sides containing regions.

Available EnSight results formats:

- EnSight Gold Case files (`*.case`)
- EnSight Dynamic Visualization Store (DVS) files (`*.dvs`)

Note:

This documentation assumes that you have already installed and configured EnSight as described in the Necessary Prerequisites section of the *Getting Started with Ansys EnSight* guide.

For more information, see:

Types of EnSight Postprocessing Files
Postprocessing with EnSight Gold Results
Postprocessing with EnSight Live Visualization Results
Setting Up the EnSight Interface
Working with Variables in EnSight
EnSight Quick Reference

Types of EnSight Postprocessing Files

The EnSight-formatted postprocessing files generated by System Coupling are summarized below:

Command files:

A single command file (`*.enc`) is written for the coupled analysis according to the type of results being generated for the run (EnSight Gold or EnSight DVS).

It contains all the data for the analysis — that is, it references all the relevant output files, including the case files for all processors used to execute the solution.

Use this file to load coupling results from System Coupling's GUI or CLI or when starting EnSight from the command line.

EnSight Gold Case files:

An EnSight Gold Case file (`Results*.case`) is written for each process that System Coupling spawns for the analysis. When multiple processes are used, file names are indexed to ensure uniqueness.

Each Case file contains only the data generated by the associated process.

Use these files after the simulation is complete to open to open coupling results in an existing instance of EnSight.

Dynamic Visualization Store (DVS) files:

When EnSight's live visualization capabilities are enabled, the Dynamic Visualization Store (DVS) user-defined writer exports the co-simulation's active variables to DVS-formatted files (`*.dvs`).

Geometry data files:

Geometry data files have a `.geo` extension. Geometry results are written for each restart point and at the end of the run.

Variable data files:

Variable data files have a `.dat` extension. For each variable, coupling step results are written for each restart point and at the end of the run.

Multi-step data files:

Filenames file:

EnSight's Filenames file is named `Results.filenames`. This file tells EnSight how to complete the filenames.

Time file:

EnSight's Time file is named `Results.time`. This file provides the time (in seconds) for transient runs.

For more detailed information on these files, see Ansys EnSight Results Files (p. 353).

Postprocessing with EnSight Gold Results

During execution of a coupled analysis, System Coupling generates results files in the Ansys EnSight Gold case file format and writes them to the **SyC/Results** folder in the co-simulation working directory.

By default, System Coupling automatically produces EnSight results files at analysis initialization, after each restart point, and at the end of the run. The subsequent postprocessing files are generated at the same frequency as restart points, as defined by the `OutputControl.Option` setting (which itself defaults to generating a single restart point at the end of the coupling run).

For more information on EnSight Gold results, see:

Changing Settings for Auto-Generating EnSight Gold Results

Generating EnSight Gold Results on Demand

EnSight Gold Postprocessing for Restarts

Loading EnSight Gold Results into EnSight

Reviewing Results in EnSight

Changing Settings for Auto-Generating EnSight Gold Results

Settings defined under `OutputControl.Results` control how and when EnSight postprocessing files are auto-generated. Change setting values as described below:

Changing Results-Generation Settings in the GUI

In the GUI's **Outline** tree under **Setup**, expand **Output Control** and click **Results**.

Edit setting values below in the **Properties** pane.

Changing Results-Generation Settings in the CLI

In the CLI, change setting values using the method shown in the example below.

```
>>> DatamodelRoot().OutputControl.Results.Option = 'StepInterval'
>>> DatamodelRoot().OutputControl.Results.OutputFrequency = 2
```

Generating EnSight Gold Results on Demand

You can generate EnSight Gold results on-demand from the CLI by running the `WriteEnSight()` command, providing the name of the file to be generated as an argument.

```
>>> WriteEnSight(FileName='EnSightResults')
```

EnSight Gold Postprocessing for Restarts

Retaining existing time step results when restarting a run

When a coupled analysis is restarted, System Coupling reads the contents of the **Results** folder and removes the following files:

- Results files for time steps beyond the initial step of the restarted run
- EnSight Command, Case, Filenames, and Time files

To postprocess time steps beyond the step at which the restarted run was opened, ensure that you either back up or rename the **Results** directory before reopening the case in System Coupling.

Generating results when a participant does not support restarts

When a coupling participant does not support restarts, EnSight results are generated for the analysis as follows:

- If `OutputControl.Results.Option` is set to the default value of ***`Program Controlled`***, then results are generated only at the end of the analysis, regardless of the restart settings specified by `OutputControl.Option`.

- If `OutputControl.Results.Option` is set to a non-default value, then results are generated as specified by the selected value.

Loading EnSight Gold Results into EnSight

Once results are generated for a coupled analysis, there are multiple ways to load them into EnSight for postprocessing. You can load System Coupling's EnSight Gold-formatted results into EnSight either when starting a new instance of EnSight or by opening them in an existing instance of EnSight.

For more information, see:

Loading EnSight Gold Results from the GUI

Loading EnSight Gold Results from the CLI

Loading EnSight Gold Results with EnSight Startup

Opening EnSight Gold Results in an Existing Instance of EnSight

---

Note:

Ansys recommends loading results into EnSight from System Coupling's GUI or CLI.

Loading EnSight Gold Results from the GUI

To load results in the GUI, use one of the following options:

- Right-click the **Solution** branch and select **Open Results in EnSight**.
- From the **Menu Bar**, selelct **Solution** → **Open Results in EnSight**.
- On the **Toolbar**, click 

.

While the results are being opened, a confirmation message is shown on the **Command Console** tab.

The **EnSight** GUI opens with the coupling results already loaded.

After the results are loaded, you may close the **System Coupling** GUI.

Loading EnSight Gold Results from the CLI

In System Coupling's CLI, run the `OpenResultsInEnSight()` command, as shown below:

```
>>> OpenResultsInEnSight()
```

A confirmation message is shown, indicating that the results are being opened. The **EnSight** GUI opens with the coupling results already loaded.

After the results are loaded, you may close System Coupling's CLI.

Loading EnSight Gold Results with EnSight Startup

When starting EnSight from the command line, you can load System Coupling's EnSight Gold-formatted results by appending command-line arguments to EnSight's executable. When loading results by this method, Ansys recommends that you use the EnSight Command file (`Results.enc`) generated for the coupled analysis.

To load results when starting EnSight, perform the following steps:

1. Open a command-line interface for your operating system in the **SyC/Results** folder for the coupled analysis.

   Note:

   If you open the command-line interface in another folder, you must use EnSight's `--cwd` command-line argument to set the path of the **Results** folder as EnSight's working directory. Simply opening the command prompt and specifying the path to the `Results.enc` file will not work.

2. Execute the platform-appropriate command, using EnSight's `-p` command-line argument to specify the name of the EnSight Command file to be played when EnSight is opened, as shown in the examples below.

   - **Windows**

     ```
     "%AWP_ROOT251%\CEI\bin\ensight.bat" -p Results.enc
     ```

   - **Linux**

     ```
     $ "$AWP_ROOT251/CEI/bin/ensight" -p Results.enc
     ```

   The **EnSight** GUI opens with the coupling results already loaded.

For more information about the `-p` argument, see Playing a Command File on Startup in the *Ansys EnSight How-To Manual*.

For information on other EnSight's command-line arguments, see Basic Usage in the Command Line Start-up Options section of the *Ansys EnSight How-To Manual*.

Opening EnSight Gold Results in an Existing Instance of EnSight

You can open System Coupling's EnSight Gold-formatted results in an instance of EnSight that is already open. When loading results using this method, Ansys recommends that you use the EnSight Gold Case files (`.case`) generated for the coupled analysis.

To open results in an existing instance of EnSight, perform the following steps:

1. Select **File > Open**.

   The **Open** dialog opens.

2. On the **Open** dialog:

   a. Ensure that **Replace data for current case** is selected.

   b. Click **OK** to close the dialog.

3. Use the **Look in** field to navigate to the **SyC/Results** folder for the coupled analysis.

4. Select **Multiple file interface**.

   The interface for loading a list of input files is shown below.

5. Locate and multi-select the `Results_<#>.case` files.

   Note:

   Ensure that all case files are selected. There is one Case file for each System Coupling process in the coupled analysis.

6. Click **Add to list**.

   The corresponding field is populated with the path to each of the selected files.

7. Click **Load all parts**.

The dialog closes and the results corresponding to the selected Case files are loaded into EnSight.

Reviewing Results in EnSight

When co-simulation results are loaded into EnSight, results are populated to the user interface as described in the following sections:

Parts

Geometry Transformations and Instancing in EnSight Results

Variables

Graphics Window

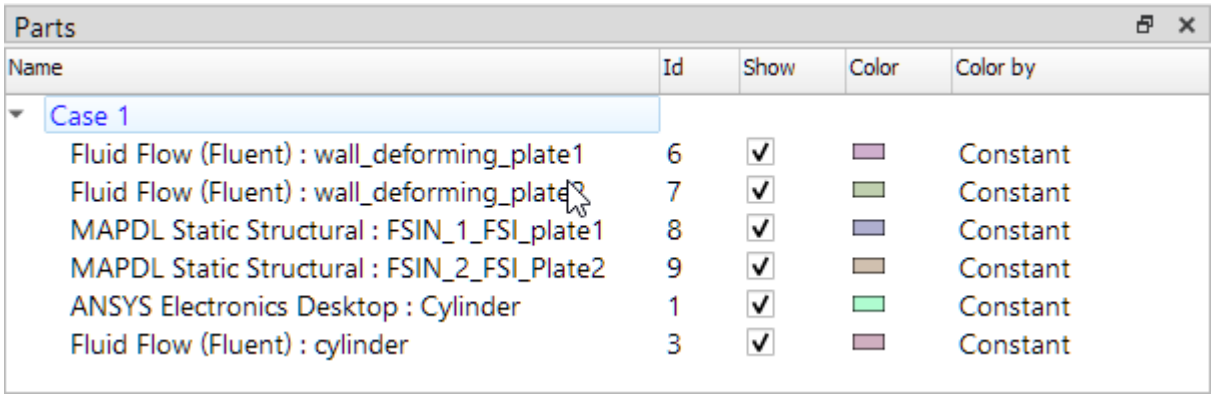
Note:

System Coupling does not enforce uniqueness of display names, but for postprocessing, modifies the variable names in the EnSight-formatted results to conform to EnSight naming requirements. These naming modifications are applied only to the EnSight-formatted results. Object display names are unchanged in System Coupling's other outputs.

Parts

The **Parts** pane is a table populated with a list of model parts (that is, coupling regions) based on the mesh and associated surfaces defined in the coupled analysis.

Figure 22: EnSight's Parts pane



Each coupled analysis is loaded into EnSight as a case, named **Case** and with an index indicating its order in the loading sequence.

Under each **Case** branch, there is a list of all regions with meshes involved in data transfers.

Each region/part is listed by its display name with a prefix indicating the display name of the corresponding participant and concatenated with additional information, as needed, to ensure that it has a unique name. Parts are named according to the following convention:

<DisplayName> : <RegionDisplayName_#>

where:

"DisplayName" is the participant's DisplayName defined in System Coupling

There is a space before and after the colon.

"#" is an optional index added when there are duplicate region display names for the same participant (for example, *HeatFlow_1* and *HeatFlow_2*).

Note:

These part/region name modifications are applied only to the EnSight-formatted results. Region display names EnSight unchanged in System Coupling's other outputs. When display names are changed for EnSight output, this information is written to the Transcript after the first completed coupling step. For more information, see EnSight Output Display Name Modifications (p. 339).

For each coupling region, columns show the following information:

- **Id**: Part number or part identifier (as designated in the results file or assigned by EnSight).
- **Show**: When selected, the part is shown in the graphics window (default for all parts on the list).

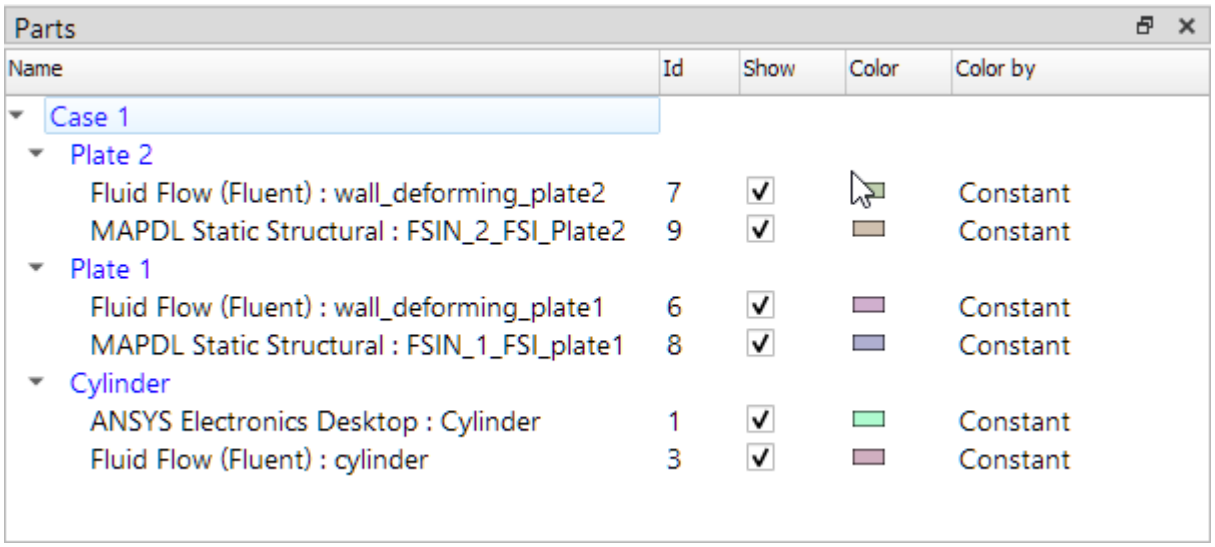
- **Color**: Color used for the region in the graphics window.
- **Color by**: Variable that determines the part coloring in the graphics window.

You can work with regions as follows:

- To change the columns shown in the table, right-click the header row and select **Customize**.
- To change the column values for a region, right-click its row and use the available context options.
- To change the attributes of a region, double-click its row and change the desired values in the **Edit Attributes for Tagged Model Part** dialog that opens.

For convenience, you may group regions according to analysis by multi-selecting the involved regions, right-clicking, and selecting **Add group/move to group**. In the image below, the groups were created in the order the analyses will be addressed in the tutorial.

Figure 23: EnSight's Parts pane with grouped regions



Geometry Transformations and Instancing in EnSight Results

When results with geometry transformations and/or instancing are written to an EnSight Results file, each reference frame or instance is written as a separate part/region.

Transformation and instancing parts are named according to following convention:

<PartName>_<#>

where:

"PartName" is the name of part/region defined in EnSight (as described in the Parts (p. 214) section)

"#" is the number of the reference frame or instance

Note:

These name modifications for transformation and instancing objects are applied only to the EnSight-formatted results. Display names for reference frames and instances are unchanged in System Coupling's other outputs. When display names are changed for EnSight output, this information is written to the Transcript after the first completed coupling step. For more information, see EnSight Output Display Name Modifications (p. 339).

Note that for a given coupling region, only one orientation and one instance can be shown at a time. For regions involved in multiple coupling interfaces, transformations and instances are displayed as follows:

- **Transformations:**

  If a region is involved in multiple coupling interfaces with dissimilar transformations, then it is shown in the `GlobalReferenceFrame`. To avoid region-orientation disparities in post-processing output, Ansys recommends that you set up the analysis such that participant regions are transformed into consistent reference frames. For example, you might define the reference frame orientation of each coupling interface such that the geometries are in their actual real-world physical orientation.

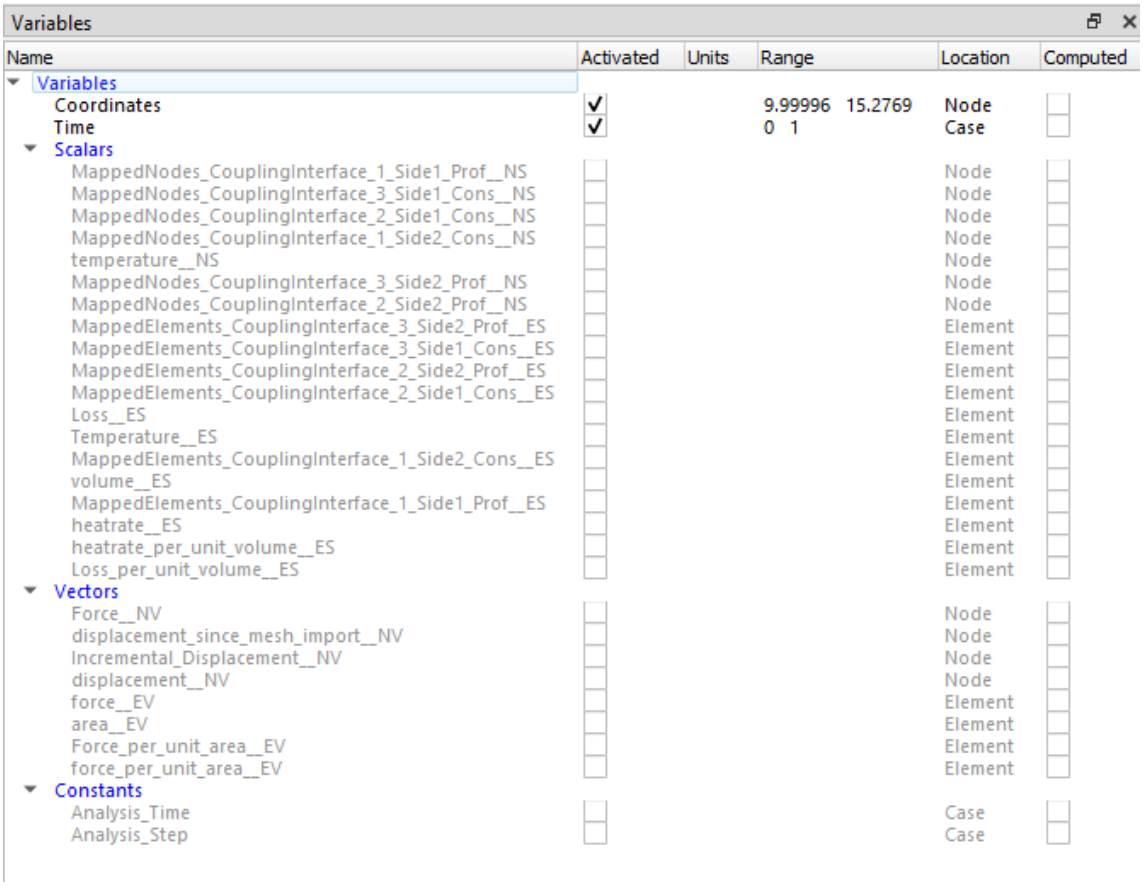
- **Instances:**

  If different instances are defined for a region that is involved in multiple coupling interfaces, then the reference instance of the region is shown. To avoid region-instancing disparities in postprocessing output, Ansys recommends that you set up the analysis such that the same instance is defined for all occurrences of a given participant region.

Variables

The **Variables** pane is a table populated with variables defined in the coupled analysis data files.

Figure 24: EnSight's Variables pane



Coordinate and **Time** variables are activated by default. For each of these variables, columns show values for units (when defined) and range. For mapping statistics variables, the number of source/target values used in the mapping is also shown.

Other variables are grouped by type as **Scalars**, **Vectors**, or **Constants**. Under each of these categories, there is a list of variables relevant to the analysis — that is, variables that are involved in data transfers.

Information on each type of variable is provided in Working with Variables in EnSight (p. 224).

Graphics Window

The **Graphics Window** shows the regions selected in the **Parts** list. Operations can be performed on regions shown in the window via the right-click context menus for regions and for the viewport itself.

To toggle between showing and hiding a region in the **Graphics Window**, select or clear its **Show** check box in the **Parts** pane.

Postprocessing with EnSight Live Visualization Results

System Coupling allows you to view your results dynamically during a co-simulation by sending results (both mesh and solution data) to either the embedded viewer (p. 93) or the EnSight GUI via EnSight's Dynamic Visualization Store (DVS) results files.

Live visualization provides a useful way to verify your co-simulation setup, allowing you to identify areas to be adjusted early in the run, rather than having to wait until the full simulation is complete.

Note:

Note: Currently, the following are not supported for live visualization:

- Multiple live visualization objects
- Complex numbers
- Instances

For more information, see:

Adding a Live Visualization Object

Enabling Live Visualization

Selecting the Live Visualization Viewer

Running a Co-Simulation with Live Visualization

Activating the EnSight DVS Reader

Adding a Live Visualization Object

To use live visualization during a co-simulation, you must first create a live visualization object.

Note:

Currently, System Coupling supports only a single live visualization object. Creation of a second object is blocked in the GUI. Creation of multiple additional objects from the CLI is possible but will result in an error.

Adding a Live Visualization object in the GUI

In the **Outline**, right-click the **OutputControl** branch and select **Add Live Visualization**.

A **Live Visualization** branch is added to the **Output Control** branch, with the new live visualization instance object defined underneath it.

Note:

Currently, it is not possible to specify a live visualization object name in the GUI. You can, however, rename existing objects in the CLI.

Adding a Live Visualization object in the CLI

Add the live visualization object by specifying an object name and an `Option` setting. Note that you may assign the object any name that conforms to System Coupling's naming conventions.

```
DatamodelRoot().OutputControl.LiveVisualization['My_FSI_LiveVisualization'].Option = "EveryStep"
```

Enabling Live Visualization

By default, the `Option` setting for a new live visualization object is set to *`ProgramControlled`*. Live visualization is disabled with a setting of *`ProgramControlled`* or *`Off`*, so it simply entails changing this setting to one of the other values.

Enabling Live Visualization in the GUI

1. Under **Output Control | Live Visualization**, click the live visualization object.
2. Change **Option** to one of the following values, based on the type of analysis:
   - **LastStep** / **LastIteration**
   - **EveryStep** / **EveryIteration**
   - **StepInterval** / **IterationInterval** (if selected, specify the **Output Frequency** as usual)

Enabling Live Visualization in the CLI

Based on the type of analysis, set `Option` to a value of *`LastStep`* / *`LastIteration`*, *`EveryStep`* / *`EveryIteration`*, or *`StepInterval`* / *`IterationInterval`* (with a specified `OutputFrequency`).

```
DatamodelRoot().OutputControl.LiveVisualization['LiveVisualization1'].Option = "LastStep"
```

Selecting the Live Visualization Viewer

You can see and interact with the live visualization results in one of two ways, the embedded viewer available in the System Coupling GUI Viewer (p. 93) tab, or through the EnSight GUI. You must manually select which option to use.

Selecting the Viewer in the GUI

1. Under **Output Control | Live Visualization**, click the live visualization object.
2. Change **Show Viewer** to one of the following values:
   - **True** (default)

     Live visualization is shown in the embedded viewer.
   - **False**

     Live visualization is shown in the EnSight GUI.

Enabling Live Visualization in the CLI

Based on preference, set ShowViewer to a value of *True* (embedded viewer) or *False* (EnSight GUI).

```
DatamodelRoot().OutputControl.LiveVisualization['LiveVisualization1'].ShowViewer = "True"
```

Running a Co-Simulation with Live Visualization

When you run a co-simulation case with live visualization enabled, the embedded viewer or the EnSight GUI opens automatically and displays the results as they become available. No other action is necessary on your part.

During the run, the following folders and files are written to the **SyC** directory:

Run type	Folders	Files
Initial run	LV_<*objectname*>	LV_<*objectname*>.enc LV_<*objectname*>.dvs
Restarted run	LV_<*objectname*>_restart<*#*>	LV_<*objectname*>_restart<*#*>.enc LV_<*objectname*>_restart<*#*>.dvs

Activating the EnSight DVS Reader

To use live visualization for the first time in a new version of EnSight, you must first activate the DVS reader, as follows:

1. In EnSight, select **Edit > Preferences**.
2. In the **Preference categories** list, select **Data**.
3. Under **Data preferences**:
   a. In the **Select below to toggle reader visibility** list, locate **DVS** and ensure that it is turned on—that is, marked with an asterisk ("*****").
   b. Click **Save to preference file**.
4. Click **Close**.

The DVS reader is now activated for the current version of EnSight.

Note:

You must repeat this process each time you upgrade to a new EnSight version.

Setting Up the EnSight Interface

To optimize the viewing of System Coupling results, set up the EnSight interface as described in the following sections:

Setting Up the Graphics Window

Set up the **Graphics Window** to provide the most effective display of coupling results, as follows:

1. Use the **View** menu to turn off the following settings:

   - Click **Perspective** to turn off perspective.

     Click **Highlight selected parts** to turn off highlighting.

2. Turn on the mesh for all regions involved in the analysis.

   a. In the **Parts** pane, multi-select all the participant regions.

   b. In the Quick Action icon bar above the **Graphics Window**, click the *Part element settings* icon () and select **3D border, 2D full**.

   c. In the Tools icon bar at the bottom of the screen, click the **Overlay hidden lines icon** ().

      The **Hidden line overlay color** dialog opens.

   d. Select **Specify line overlay color**.

   e. Click **OK** to close the dialog.

Setting Up the Viewports

Set up the viewports so results on each side of the coupling interface are visible at the same time. To do so, perform the following steps:

1. Split the **Graphics Window** into two viewports.

   Right-click inside the window and select **Viewports > 2 vertical**.

Note:

- If you only need to show results on one side of the interface, you may skip this step and keep the single viewport.

- Choose the viewport orientation best suited to your model. For example, you would likely use horizontal viewports for a geometry that is elongated along the x axis.

2. Right-click inside the **Graphics Window** and select **Viewports > Link All**.

   This links the viewports together so the same view is shown in each.

3. Limit the display of each participant's interfaces to a single viewport. For each participant:

   a. In the **Parts** pane, multi-select all regions for that participant.

   b. In the Quick Action icon bar above the **Graphics Window**, click the *Visibility per viewport* icon ().

      The **Part viewport visibility** dialog opens. Both squares are green, indicating that the participant's interfaces are visible in both viewports.

   c. Click a green square.

      If you click the square on the left, the right side of the dialog is now black, indicating that the participant's interfaces are now shown only in the left viewport. The opposite occurs if you click the right square.

Turning Off the Continuous Palette Setting

When working with System Coupling results in EnSight, it is generally recommended that you turn off EnSight's continuous palette setting (which is enabled by default).

Coupling participant data may be stored on either nodal or elemental locations. When data are stored on elements, there is a constant value per element. When EnSight's continuous palette setting is enabled, element data are scattered (averaged) to the element's nodes and then presented as though they were nodal data. When the continuous palette setting is disabled, each element is displayed with its constant value, which provides a better representation of how System Coupling has actually mapped the data.

Note:

Because nodal data is not scattered, disabling the continuous palette setting has no effect when data is stored on nodes.

To turn off continuous palette coloration, perform the following steps:

1. Select **Edit > Preferences**.

   The **Preferences** dialog opens.

2. Under **Preference categories**, select **Color Palettes**.

Color palette/legend preferences are shown below.

3. Clear the **Use continuous palette for per-element variables** check box.

Continuous coloration is now disabled when data on elements are displayed.

Figure 25: Continuous vs non-continuous palette coloration for element-based mapping variables

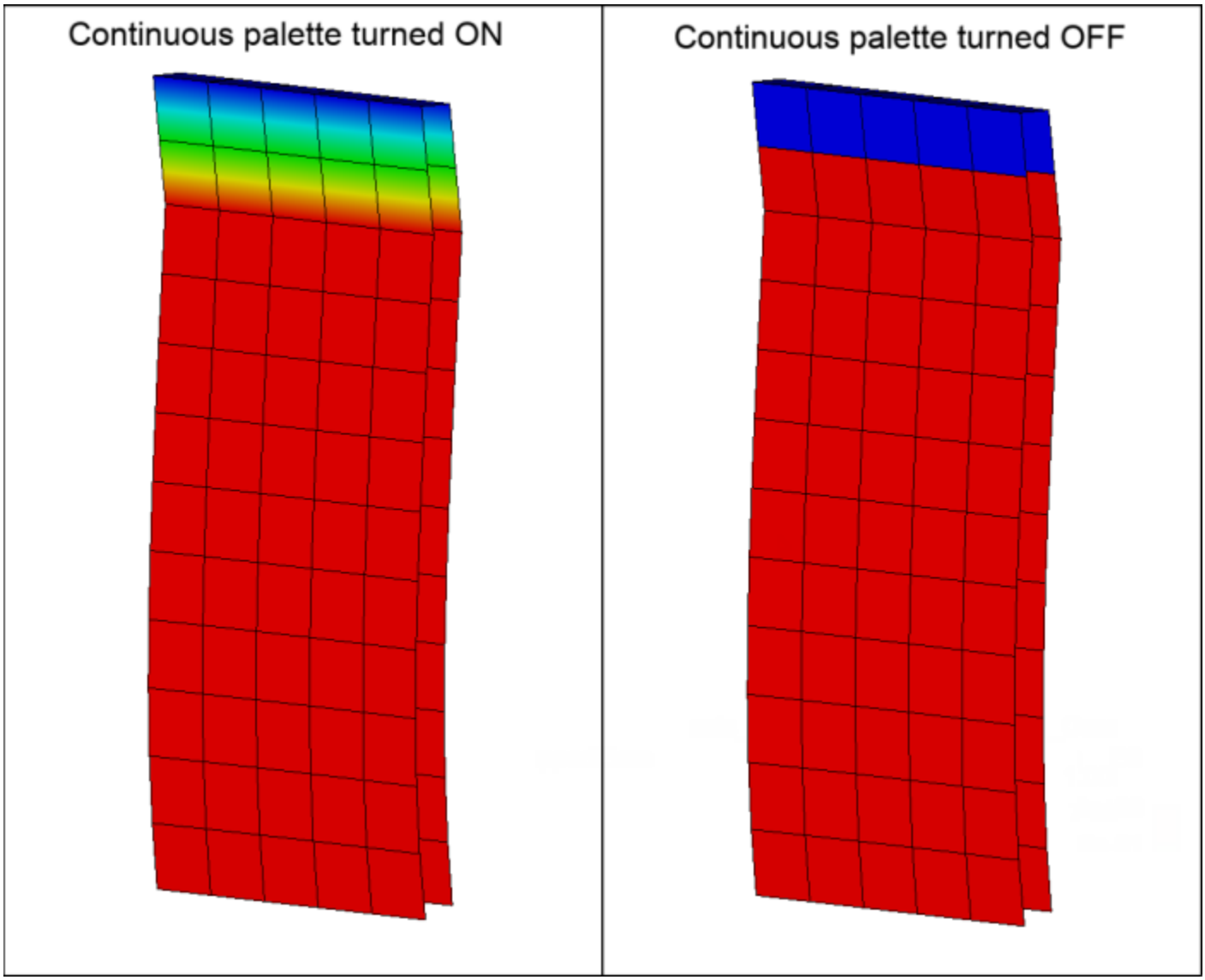
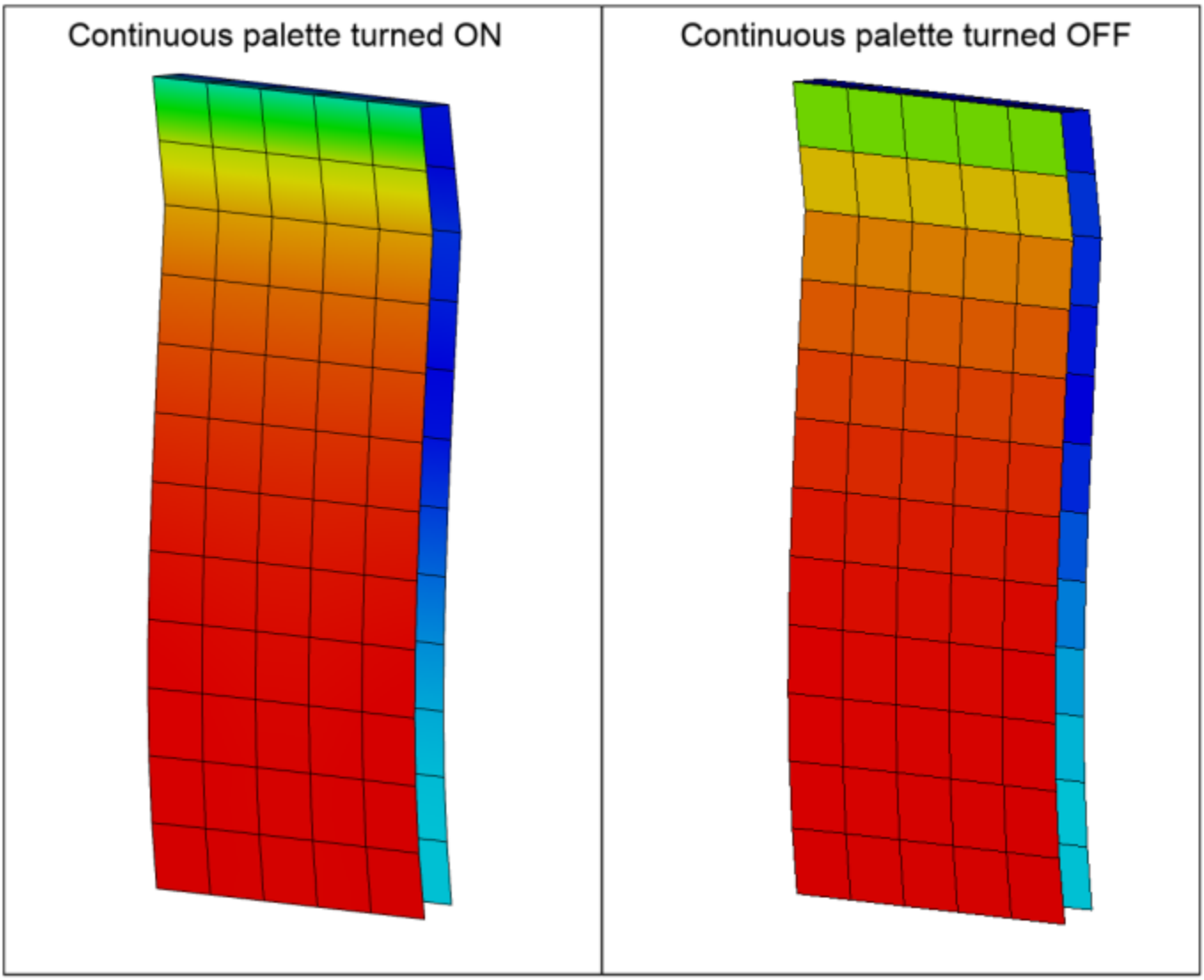


Figure 26: Continuous vs non-continuous palette coloration for per-element data transfer variables



Working with Variables in EnSight

For information on working with System Coupling variables in EnSight, review the following sections:

Variable Naming Conventions

Selecting Variables

Using Mapping Variables

Visualizing Variable Data

Adjusting Plot Palette Ranges

Variable Naming Conventions

Each variable is listed by its display name, concatenated with additional information, as needed, to ensure that it has a unique name.

Naming conventions vary according to the type of variable.

Variables

***<VariableDisplayName_#>*__source__<[*N*|*E*][*S*|*V*]>-->_<[*real*|*imag*]>-->**

Where:

- "***#***" is an optional index added when there are duplicate variable display names for the same participant (for example, **FluentFluidFlow_1** and **FluentFluidFlow_2**)
- The variable display name (with index, when applicable) is followed by two underscores
- "**source**" is an optional indicator added when the source variable has been defined by an expression
- "***N***" or "***E***" indicates the variable data **Location** (**N**ode or **E**lement)
- "***S***" or "***V***" indicates the variable **Quantity Type** (**S**calar or **V**ector)
- "***real***" or "***imag***" indicates the **Data Type** (Real or Imaginary) for complex variable components

---

Note:

- These variable name modifications are applied only to the EnSight-formatted results. Variable display names are unchanged in System Coupling's other outputs. When variable display names are changed for EnSight output, this information is written to the Transcript after the first completed coupling step. For more information, see EnSight Output Display Name Modifications (p. 339).
- System Coupling updates nodal coordinates and related mesh quantities at every coupling iteration after displacement is transferred. To facilitate EnSight postprocessing, System Coupling provides a **displacement-since-mesh-import__NV** variable. This variable accumulates the displacement calculations since the last mesh import (typically, at the beginning of the analysis or at the last restart) for all incremental displacement variables defined on the coupling interface.
- System Coupling creates an **area__EV** variable for the area vectors of all surface face zones in the analysis. Post-processing this field can be helpful in understanding mapping behavior and to choose the best `FaceAlignment` setting for advanced situations such as one-sided shells.

Mapping Statistics Variables

Source Side

Used<*N* | *E*>_<*interfaceName*>_<*targetVariableName*>

Where "***N***" or "***E***" indicates the variable data location (**N**ode or **E**lement)

Target Side

Mapped<*N* | *E*>_<*interfaceName*>_<*targetVariableName*>

Where "***N***" or "***E***" indicates the variable data location (**N**ode or **E**lement)

These variables must be activated (loaded into memory) either explicitly by selection of the corresponding **Activated** check box, or by being called by an operation that requires it.

Selecting Variables

The EnSight variables used to view data transfer data — whether mapping diagnostics or transferred values — on a given interface side should be determined by **the corresponding participant's data location for that data transfer** — that is, whether the participant stores data for that quantity type on nodes or elements.

Start out by finding the **Location** value for each participant variable involved in a data transfer. This information is available in the **Coupling Participants** section of the Transcript's **Summary of Coupling Setup**. Data locations are determined by the coupling participants and cannot be changed. Make note of these values, if needed, because you will use this information later to select the variables that match each participant's data location for a given transfer.

The example below shows Maxwell's participant information, with its data transfer variables and corresponding data locations highlighted in yellow.

Figure 27: Maxwell's data transfer variables and data locations

```
+=============================================================================+
|                          Coupling Participants (3)                          |
+=============================================================================+
|                                                                             |
| Participant: ANSYS Electronics Desktop                                      |
|    Type :                                                          AEDT     |
|    UpdateControls:                                                          |
|       Option :                                          ProgramControlled   |
|    Region: Cylinder                                                         |
|       Topology :                                                   Volume   |
|       Input Variables :                                         Temperature |
|       Output Variables :                                           Loss     |
|    Variable: Loss                                                           |
|       Quantity Type :                                            Heat Rate  |
|       Location :                                                   Element  |
|       Tensor Type :                                                Scalar   |
|       Is Extensive :                                               True     |
|       Variable Name :                                              Loss     |
|    Variable: Temperature                                                    |
|       Quantity Type :                                           Temperature |
|       Location :                                                   Element  |
|       Tensor Type :                                                Scalar   |
|       Is Extensive :                                               False    |
|       Variable Name :                                           Temperature |
|                                                                             |
+-----------------------------------------------------------------------------+
```

For data transfers of extensive variables (such as Force or Heat Rate), the amount of the conserved quantity in each mesh element will vary with the sizes of those elements. To facilitate comparison of source and target values on meshes with different element sizes, you should always view the corresponding intensive variables — that is, **per-unit-area** or **per-unit-volume** on surface or volume regions, respectively.

Using Mapping Variables

In System Coupling analyses, all data transfers are accomplished via **mapping**, which is the process of calculating data at locations on the target mesh using data from locations on the source mesh. Mapping quality depends upon factors such as the proportion of the target mesh locations that receive data directly from source mesh locations, and the proportion of source mesh locations that are used in those target data calculations. It is important to have high-quality mapping between the source and target meshes on a coupling interface because this will affect the quality of the data transfers.

Note:

To assess mapping quality, you should be familiar with System Coupling's mapping process and the types of mapping used in different scenarios. For more detailed information, see System Coupling's Mapping Capabilities (p. 282).

For more information, see:

Mapping Type Applicability

Because mapping weights are generated according to mapping type, a given mapping variable covers all data transfers of the specified mapping type that are sent to the specified interface side. For example, if multiple conservative data transfers are sent to the same side of the same interface, the same mapping variable would be applicable to all of them.

Also, a given mapping variable may be applicable to multiple data transfers. For example, a Heat Transfer Coefficient variable and a Convection Reference Temperature variable are transferred as a pair to the same target side on an interface. The same mapping variable will be applicable for these transfers since they both use profile-preserving mapping.

For more information about the mapping type used for various quantities, see Participant Variables and Quantity Types Supported by System Coupling (p. 28).

Mapping Variable Display

To apply a mapping variable, drag it from the **Variables** pane and drop it on the relevant interface side(s) in the **Graphics Window**. When evaluating mapping quality, review both the mapped (used) and unmapped (unused) locations on the source and target sides of the interface.

As noted previously, mapping variables indicate how closely the target side of the interface reproduces the source side. Mapping variables are unitless and are displayed in EnSight as follows:

- Red (a value of **1**) indicates that the location is mapped. Source locations have been successfully mapped to target locations and the source data are being used by the target.
- Blue (a value of **0**) indicates that the location is not mapped. Source locations have not been successfully mapped to target locations, so the source data are not being used by the target.

To better reflect System Coupling's mapping values, you should configure EnSight as described in Turning Off the Continuous Palette Setting (p. 222).

Visualizing Variable Data

To visualize results, you will show variable data on participant regions. One variable may be shown simultaneously on multiple regions, but each region can show only one variable at a time.

To show variable data on a participant region, drag the variable from the **Variables** pane and drop it onto the corresponding interface side(s) in the **Graphics Window.**

Note:

See the EnSight documentation for other methods applying variables to participant regions.

When the variable is applied:

- The variable is activated. In the **Variables** pane:
  - The variable name and its parameters are shown as enabled (in black text).
  - The variable's **Activated** check is box selected.
  - The **Range** parameter is updated with the minimum and maximum values for the range.
- Each affected region is colored by the variable.
  - In the **Parts** pane, the region's **Color by** parameter is updated to show the variable name.
  - In the **Graphics Window**, variable values are shown on each region and a legend is added to the window.

Adjusting Plot Palette Ranges

When comparing data transfer values on both sides of an interface, ensure that plots have palette ranges that are consistent with one another. The goal is to select a range that shows a smooth representation of the full range on both sides of the interface.

Note:

Adjusting palette ranges for consistency is effective only for data transfers between like region topologies (that is, transfers between two surface regions or between two

volume regions). For data transfers between unlike region topologies, matching ranges will result in poor plots for both models.

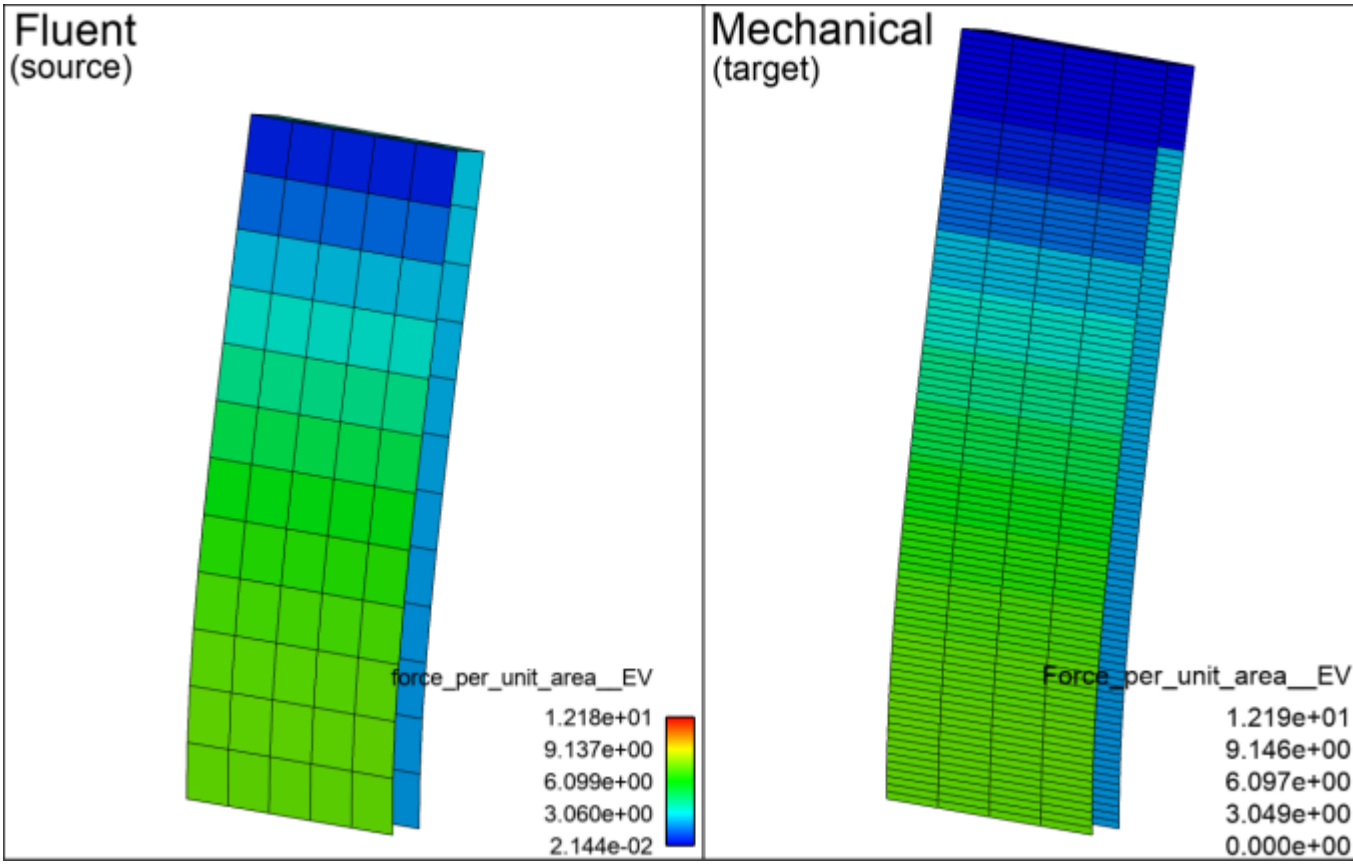
In a co-simulation with 2D-3D transfers, it is preferable to plot the transferred quantity using a ***per-unit-volume*** variable. For an example, see Visualize Loss Results and Visualize Losses Per-Unit-Volume in the *Permanent Magnet Electric Motor Co-Simulation* tutorial.

To adjust the palette ranges for data transfer plots, perform the following steps:

1. Identify any issues with the representation of the range.

   In the example image below, note that the plot palette ranges are dissimilar. Also, neither plot has mesh locations that show values at the upper end of their ranges.

   Figure 28: Force Per-Unit-Area data transfer plots with min-max palette ranges



2. Identify a range that is appropriate for both plots.

3. Adjust the upper and lower boundary of both ranges as appropriate. For each side of the interface, perform the following steps:

   a. Right-click the palette bar and select **Palette > Edit palette**.

      The **Palette Editor** dialog opens.

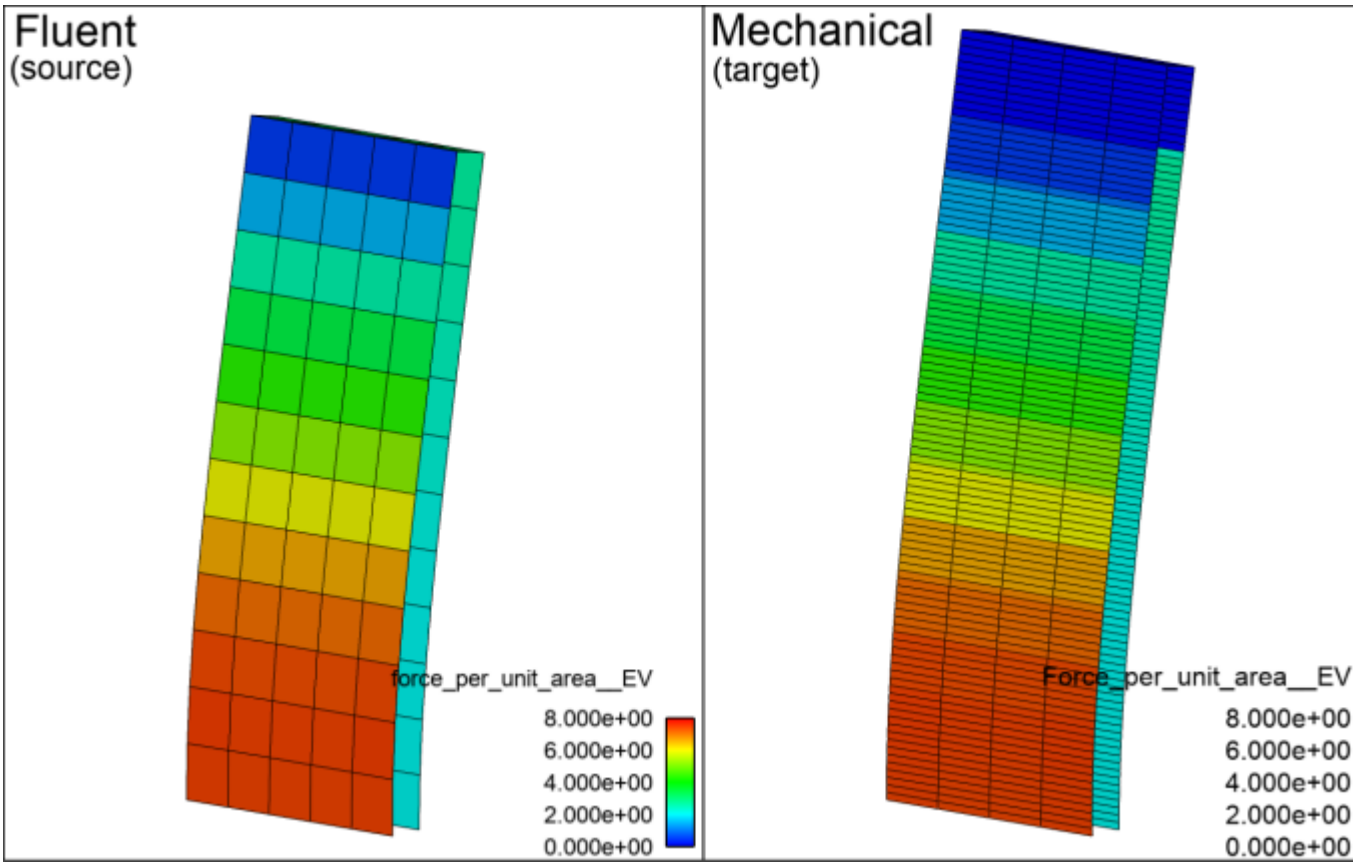
b. In the field to the right of **Range Used**, type in the upper range number.

c. In the field to the left of **Range Used**, type in the lower range number.

d. Click **Close** and then **Close** again to exit both dialogs.

Both plots now share a smooth representation of the full range of values, as shown in the image below.

Figure 29: Force Per-Unit-Area data transfer plots with adjusted palette ranges



EnSight Quick Reference

The following sections provide instructions on EnSight postprocessing tasks that are commonly used in the postprocessing of co-simulation results:

Adding a Text Annotation

Running a Simple Animation of Solution Time Steps

Adding a Simple Coordinate Probe

Adding a Time Annotation

Plotting Nodal Displacement as a Function of Time

Adding Force Vector Arrows

Adding a Text Annotation

Add a text annotation to a viewport in EnSight.

1. Right-click inside the viewport and select **Quick annotation > Text**.

   The **Create/edit annotation (text)** dialog opens.

2. In the text box, type in the text you to be displayed.

3. Adjust the size of the text annotation as needed by editing the value of the **Size** field.

4. Click **Close**.

5. Left-click the annotation to select it.

6. Holding down the left mouse-button, drag the annotation it to the desired location.

Running a Simple Animation of Solution Time Steps

Run a simple animation of solution time steps in EnSight.

1. In the **Solution Time Player** (that is, the **Time** pane), note the value of the **Cur** field.

   The player defaults to using discrete time steps, with the **Cur** field (the current time step) defaulting to the last time step in the solution.

2. Set the player back to the first time step setting the **Cur** field to **0**.

3. Leave the player to its default behavior of looping the animation ().

4. Run the animation by clicking *Play Forward* ().

Adding a Simple Coordinate Probe

Add a simple probe to an animation in EnSight.

1. In the feature icon bar, click *Interactive probe query* ().

   The **Create/edit probe query** dialog opens.

2. On the **Probe create** tab:

   a. Under **Which variables?**, select **Coordinates**.

   b. Set **Query** to **Surface pick**.

   c. For **Probe count**, type in the number of probes to be created.

   d. Set the **Search** fields to **Closest node** and **Pick (use 'p')**.

The node will be placed on the node closest to the location you select using the '**p**' key on your keyboard.

e. Add the probes, performing the following steps for each:

i. Left-click the model where you want to add the probe.

ii. Press the '**p**' key on your keyboard.

The probes have been added to the model.

Note:

On the **Create/edit probe query** dialog:

- To view probe details, click **Display results table**.
- To change the appearance of probes, use the settings on the **Display style** tab.
- To add additional probes, increase the **Probe count**.
- To remove the probes, disable the probe query by setting **Query** to **None**.

Adding a Time Annotation

Add a time annotation to an animation in EnSight.

1. In the **Time** pane, click the *ellipsis* ().

   The **Solution time** dialog opens.

   Note:

   If the **Time** pane is not visible, you may display it by right-clicking the **Parts** pane header bar and selecting **Time**.

   ---

2. Click **Display time annotation**.

   The **Create/edit annotation (text)** dialog opens.

3. Adjust the size of the time annotation as needed by editing the value of the **Size** field.
4. Click **Close**.
5. Left-click the annotation to select it.
6. Holding down the left mouse-button, drag the annotation it to the desired location.

Plotting Nodal Displacement as a Function of Time

Plot nodal displacement as a function of time.

Note:

This example uses the results from the Oscillating Plate FSI Co-Simulation with Partial Setup Export from Workbench (CFX-Mechanical) tutorial.

Create the Function

1. In the **Parts** pane, select the fluid region. For this example, select **CFX : wall_deforming**.
2. In the Feature Icon bar at the top of the window, click the *Calculator* icon ().

   The **Calculator Tool Box** dialog opens.
3. Click the *Build your own functions* icon ().
4. For **Variable** name, type in **`nodal_position_x`**.
5. Create an expression.
   a. Under **Variable**, click **Coordinates**.
   b. In the calculator, click the **[X]** button and then the **+** (addition) button.
   c. Under **Variable**, click **displacement_since_mesh_import__NV**.
   d. In the calculator, click the **[X]** button.

      Each of these is added to the **Expressions** field.
6. Click the **Evaluate for selected parts** button.

   The **nodal_position_x** vector quantity is created.
7. Click **Close** to exit the dialog.

Create the Plot

1. In the Feature Icon bar at the top of the window, click the *Query* icon ().

   The **Create/Edit Query/plot** dialog opens.
2. Under **Query Creation**, make the following selections:
3. For **Sample**, select **At node over time**.

4. For **Variable 1**, select the vector quantity created in the previous step, **(N) nodal_position_x**.
5. For **Node ID**, type in **`10`**.

   This is the top corner node in the fluid setup.
6. Click the **Create query** button.

   The plot is added to the **Graphics Window**.
7. Click **Close** to exit the dialog.

You may also create a time annotation and animate the plot, as described in Adding a Time Annotation (p. 232) and Running a Simple Animation of Solution Time Steps (p. 231), respectively.

Adding Force Vector Arrows

Add force vector arrows to an animation.

Note:

This example uses the results from the Reed Valve FSI Co-Simulation with Partial Setup Export from Workbench (Fluent-Mechanical) tutorial.

---

1. In the **Parts** pane:
   a. Ensure that the appropriate force per-unit variable is applied to the region to which force is being applied.

      In the Reed Valve tutorial example, the **Force_per_unit_area__EV** variable is applied to **Transient Structural : System Coupling Region**.
   b. Select that region.
2. In the Feature Icon bar at the top of the window, click the *Vector Arrows* icon ().

   The **Create Vector Arrows using default attributes** dialog opens.
3. In the blank field near the top of the dialog, you may name the new part to be created. Type in **`Force Per-Unit-Area Vector Arrows`**.
4. Confirm that **Variable** to is set to the force per-unit vector variable — in this case, **(E) Force_per_unit_area__EV**.
5. Optionally, you may wish to adjust the display of the arrows. To do so, ensure that the **Advanced** check box is enabled and then edit the display parameters as needed. For this example:
   a. Under **Creation**, for **Scale factor** type in **`2.0000e-07`**.

b. Under **Arrow Tip**, leave **Size** set to **Proportional** and type in **`1.5000e-01`**.

6. Click the **Create with selected parts** button.

   A **Force Per-Unit-Area Vector Arrows** entry is added to the **Parts** pane.

7. Click **Close** to exit the dialog.

You can also animate the plot, as described in Running a Simple Animation of Solution Time Steps (p. 231). The arrow shows the vector of the forces applied to the body.

Using System Coupling in Workbench

This section provides a high-level overview of System Coupling in Workbench.

When running System Coupling in Workbench, you use the Workbench interface and workflow to build and execute coupled analyses.

Note:

When executing a coupled analysis within the Workbench environment, Workbench is responsible for starting the coupling participant processes that will connect to System Coupling. For System Coupling, this means that each participant's execution control `Option` is set to `ExternallyManaged`. For more information about execution control options, see `ExecutionControl` in the *System Coupling Settings and Commands Reference* documentation.

The following topics are addressed in this section:

- Overview of System Coupling in Workbench (p. 237) provides information on supported participants, data transfers possible, the rules for creating data transfers, and the **System Coupling** tab.
- Setting Up a Coupled Analysis in Workbench (p. 243) provides instructions on preparing for, creating, and changing settings for a coupled analysis.
- Running a Coupled Analysis in Workbench (p. 258) provides instructions for running, monitoring, and reviewing output for a coupled analysis.
- Common Tasks for System Coupling in Workbench (p. 267) describes how to stop and restart coupled analyses that are run in Workbench..
- Advanced Tasks for System Coupling in Workbench (p. 271) describes advanced tasks that can be used in workflows for System Coupling in Workbench — specifically, requesting debug logging output and using Workbench to set up a coupled analysis to be executed in one of System Coupling's user interfaces.

Overview of System Coupling in Workbench

This section provides a high-level overview of System Coupling in Workbench, describing the capabilities supported by the **System Coupling** component system.

Note:

Some of these capabilities, however, may not yet be supported in conjunction with other Workbench systems. For information about systems that can act as participants

in coupled analyses, see Supported Coupling Participants for System Coupling in Workbench (p. 238).

The following topics are addressed:

- Supported Coupling Participants for System Coupling in Workbench (p. 238) describes the systems that can serve as participants in a coupled analysis.
- Data Transfers for System Coupling in Workbench (p. 239) describes rules for creating data transfers.
- Steps of a Coupled Analysis in Workbench (p. 240) lists the steps involved in setting up and running a coupled analysis within Workbench.
- System Coupling Tab in Workbench (p. 241) describes the parts of the System Coupling workspace in Workbench.

Supported Coupling Participants for System Coupling in Workbench

System Coupling *participants* are systems that send and/or receive data in a coupled analysis. Supported participant systems for System Coupling in Workbench include:

CFX

- **Fluid Flow (CFX)** analysis system
- **CFX** component system

For information on using System Coupling with a CFX system, see the following sections in the *CFX-Solver Modeling Guide*:

- Coupling CFX to an External Solver: System Coupling Simulations
- Restarting CFX Analyses as Part of System Coupling

Fluent

- **Fluid Flow (Fluent)** analysis system
- **Fluent** component system

For information on using System Coupling with a Fluent system, see the following sections in the *Fluent User's Guide*:

- Performing System Coupling Simulations Using Fluent
- Restarting Fluent Analyses as Part of System Couplings

Mechanical

The following analysis systems can participate in coupled analyses:

- **Coupled Field Static**

- **Coupled Field Transient**
- **Static Structural**
- **Transient Structural**
- **Steady-State Thermal**
- **Transient Thermal**
- **Thermal Electric**

For information on using System Coupling with a Mechanical system, see the following sections in the *Mechanical User's Guide*:

- System Coupling
- Restarting Structural Mechanical Analyses as Part of System Coupling

External Data

- **External Data** component system

For information on using System Coupling with an External Data system, see the following section in the *Workbench User's Guide*:

- External Data

Note:

The following limitations are placed on the participants of a coupled analysis:

- Only two coupling participants can be connected to the **System Coupling** system at one time. However, more than one **System Coupling** system may be introduced on the same **Project Schematic**.
- Fluent can be connected with any of the other supported participants.
- Coupled analyses are not supported for **External Data** participant systems that have non-ASCII characters in the following values:
  - Display names for regions or variables
  - Data source **Identifier** (defined in the **External Data** tab's **Outline of Schematic**)
  - Data source **Data Identifier** (defined in the **External Data** tab's **Table of File**)

Data Transfers for System Coupling in Workbench

A *data transfer* is the transfer of a single quantity type in one direction between source and target regions of two participant meshes. For System Coupling in Workbench, each data transfer is defined by a variety of user interface settings (p. 251) such as **Source**, **Target**, and **Data Transfer Control**.

For more information on supported data transfers, see Participant Variables and Quantity Types Supported by System Coupling (p. 28).

For information on the data transfers that are supported for a given participant system, use the links in Supported Coupling Participants for System Coupling in Workbench (p. 238) to review participant product documentation.

When you create data transfers using the **Create Data Transfer** context option in the **System Coupling** tab's **Outline of Schematic**, the following rules apply:

- For a given target region, there can only be one source region.
- Thermal data can be transferred:
  - between Fluent and Mechanical directly through System Coupling,
  - between CFX and Mechanical directly through System Coupling, and
  - through the coupling of an **External Data** component system.
- For one-way thermal transfers, only one of the above three options above can be defined for a given pair of source and target regions.

For data transfer creation details that are generally applicable to all System Coupling contexts, see Rules for the Creation of Data Transfers (p. 281).

Steps of a Coupled Analysis in Workbench

To create and run a coupled analysis, perform the following steps:

1. **Create the analysis on the Project Schematic. (p. 243)**
   a. Add participant systems to the **Project Schematic**.
   b. Add the **System Coupling** system to the **Project Schematic**.
2. **Set up the physics for each participant system. (p. 243)**
   a. Configure the participant settings that enable a coupled analysis.
   b. Set up each participating system (generally from top-to-bottom, until you have completed all the required steps for your analysis) and perform a test solve.
3. **Connect the systems. (p. 244)**
4. **Set up the System Coupling system.**
   a. Modify settings needed to complete the coupled analysis. (p. 245)
   b. Optionally, enable expert settings (p. 256).
5. **Run the analysis. (p. 258)**
   a. Start the solution. (p. 259)

b. Monitor the solution progress. (p. 259)

6. **Optionally, stop and restart the analysis.**.

   a. Stop the analysis. (p. 267)

   b. Modify the analysis as needed. (p. 245)

   c. Restart the analysis. (p. 268)

7. **Review analysis output.**

   a. Review System Coupling's output. (p. 330)

   b. Review the participant results.

Note:

The steps vary slightly for a coupled analysis being exported from Workbench.

System Coupling Tab in Workbench

The **System Coupling** tab is the workspace where you will finish setting up and then run a coupled analysis.

To open it, edit either the **Setup** cell or the **Solution** cell of your **System Coupling** component system. The tab opens in your Workbench project, as shown in Figure 30: The System Coupling tab (p. 242).

The **System Coupling** tab has two nodes which correspond to the cells of the **System Coupling** system on the **Project Schematic**:

- ***Setup***

  Use this node to view participant, region, and variable information, and to define analysis settings and data transfers between participants.

- ***Solution***

  Use this node to solve a coupled analysis and to view solution information and charts.

Figure 30: The System Coupling tab

The following panes are all displayed by default:

Outline of Schematic:

Shows a tree with nodes relating to various parts of the coupled analysis. It has two nodes which correspond to the **System Coupling** system's cells on the **Project Schematic**, as follows:

- a **Setup** node, which you will use to complete analysis settings (p. 245), including details for participants, data transfers, and execution controls, and
- a **Solution** node, which you will use to start the solution (p. 259) and monitor its progress (p. 259) during execution.

When a node is selected in the **Outline of Schematic**, corresponding properties are shown in the **Properties** pane.

Properties:

Shows setup nodes and properties (p. 247) corresponding to the item selected in the **Outline of Schematic**.

Chart Monitor:

Shows System Coupling charts during and after the solution.

Solution Information:

Shows a text-based log (p. 332) of information output during the execution of the coupled analysis.

For more information about the tabbed panes in Workbench, see Panes within Tabs in the *Workbench User's Guide*.

Setting Up a Coupled Analysis in Workbench

To set up a coupled analysis using System Coupling in Workbench, perform the following steps:

Creating a Coupled Analysis

Completing Participant Setups

Connecting Systems on the Project Schematic

Completing the Coupling Setup

Note:

You can set up a coupled analysis in Workbench and then execute it in one of System Coupling's user interfaces. For more information, see Exporting a System Coupling Setup (p. 273) and Running an Exported System Coupling Setup (p. 190).

Creating a Coupled Analysis

To create a coupled analysis in Workbench:

1. Add individual participant systems to the Workbench **Project Schematic**.

   For example, you might:

   - Add analysis systems for two dynamic participants, such as a **Transient Structural** system for Mechanical and a **Fluid Flow (CFX)** system for CFX.
   - Add systems for a static data source and a dynamic participant, such as an **External Data** component system for an external data source and a **Fluid Flow (Fluent)** analysis system for Fluent.

2. Add a **System Coupling** component system to the **Project Schematic**.

   This system will be used to connect the participants later.

Completing Participant Setups

Complete the physics setup for each participant to enable its inclusion in the coupled analysis. For each participant, open the product and perform the following steps:

1. Complete the fluid and/or solid physics setup.

   During the coupled analysis, System Coupling manages participant solutions. However, each participant should be able to solve successfully on its own prior to being included in the coupled analysis.

   Note:

   - When using same material(s) in multiple participants, take care to ensure that common properties are consistently defined, as described in Additional Participant Setup Considerations (p. 118).

- Directory paths and object names (for interfaces, participants, regions, and variables) must be in ASCII format. Object names also cannot include forward slashes (/).

For additional suggestions on how to ensure that your coupled analysis runs smoothly, see Building Up to a Coupled Analysis (p. 393).

2. Configure coupling-specific settings to enable coupling.

   - For Mechanical, this involves creating a **System Coupling Region**.
   - For CFX and Fluent, this involves creating one or more boundary conditions.

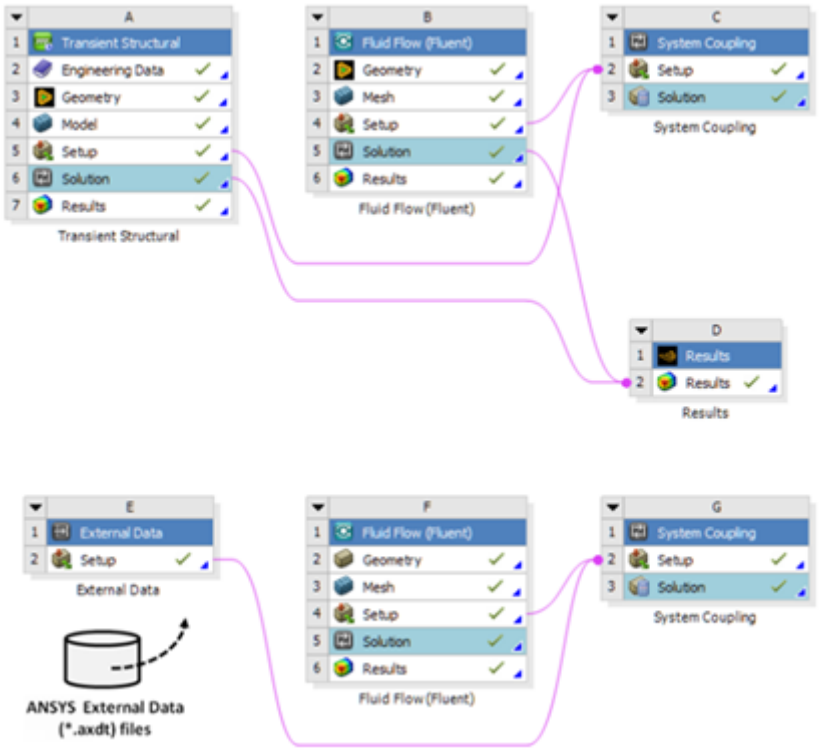
   For more information on a given participant's settings, use the links in Supported Coupling Participants for System Coupling in Workbench (p. 238) to access its documentation.

Connecting Systems on the Project Schematic

Connect the systems on the **Project Schematic** via their **Setup** cells, as shown in Figure 31: Connecting a System Coupling component system with other types of systems (p. 244).

Most participant systems with **Setup** cell connections act as co-simulation participants. The exception to this is the **External Data** system. Although a connection originates from its **Setup** cell, the **External Data** system acts as a static participant.

Figure 31: Connecting a System Coupling component system with other types of systems



The execution of analyses involving couplings between any of these participants is managed by System Coupling. For details, see Coupled Analysis Management (p. 277).

Once a participant system is connected to the **System Coupling** system, updates of the participant's **Solution** cell are disabled. All solution updates (and the execution of the respective solvers) are started automatically when System Coupling's **Solution** cell is updated. Note, however, that these updates respect all settings (for example parallel, precision, and so on) already made for them.

Note:

For remote execution of a co-simulation created in Workbench, the recommended workflow is to export the coupling setup from Workbench (p. 273) and then run the co-simulation in System Coupling's GUI or CLI (p. 190), using System Coupling's parallel processing capabilities (p. 190) to submit jobs for remote execution.

To continue using previous functionality, you may use one of the workflows described in Submitting System Coupling Jobs to Remote Solve Manager in the *Workbench User's Guide*.

Completing the Coupling Setup

Set up the System Coupling portion of the coupled analysis using the sub-nodes and corresponding properties available under the **Setup** node, which is defined in the **System Coupling** tab's **Outline**.

The state of the **Setup** node reflects the state shown by the **System Coupling** system's **Setup** cell on the **Project Schematic**. Validation of the **Setup** node depends upon the validation of the individual sub-nodes defined under it (for example, **Analysis Settings** and **Data Transfer**). If any of these sub-nodes is invalid, then the **Setup** node is also invalid, as shown by the state of *Attention Required* (?).

Tip:

If the Workbench setup will be exported for execution in one of System Coupling's user interfaces, then proceed according to the capabilities that are required for the analysis. Available capabilities are determined by the timing of the export, as described in Exporting Partial vs. Full Coupling Setups (p. 274).

For more information, see Exporting a System Coupling Setup (p. 273) and Running an Exported System Coupling Setup (p. 190).

To access System Coupling properties, right-click the **Setup** node and select **Expand All**.

Use the following **Setup** node functionality to complete your coupled analysis setup:

Setup Node Context Options

Coupling-Specific Properties

Setup Node Context Options

Use the **Setup** node's context options to perform various set-up tasks. To access available options, right-click a sub-node defined under the **Setup** node.

Available options:

Expand All / Contract All

Available for all nodes. Expands or contracts all nodes in the **Outline**.

Start/Stop highlighting linked nodes

Available for the **Setup** node. Controls whether nodes related to the selected node are highlighted in the **Outline**.

Create Data Transfer

Available for the **Data Transfers** container node. Creates a new data transfer. For details, see Data Transfer Context Options (p. 253).

Rename

Available for data transfer nodes. Renames the selected data transfer. For details, see Data Transfer Context Options (p. 253).

Duplicate

Available for data transfer nodes. Duplicates the selected data transfer. For details, see Data Transfer Context Options (p. 253).

Display Validation Failure

Available for any node that is invalid, as indicated by a state of *Attention Required* (). Displays error messages with details on validation problems.

Add Property

Available for the **Execution Control > Expert Settings** node. Allows you to select and enable an expert setting. Once enabled, the setting is displayed under **Properties** whenever the **Expert Settings** node is selected. For more information, see Expert Settings (p. 256).

Remove Property

Available for the **Execution Control > Expert Settings** node when an expert setting is enabled. Allows you to select and disable an expert setting. Once disabled, the setting is removed from **Properties**. For more information, see Expert Settings (p. 256).

Read restart points

When **Analysis Settings** is selected in the **Outline**, available for the **Coupling Initialization** property under **Properties**. Populates the list of restart points.

Useful for atypical situations such as a Workbench crash, when the restart point list may be empty even though the intermediate restart files exist on your disk. Repopulating the list of restart points allows you to restart from a previously saved point.

Export System Coupling Setup

Available for the **Setup** node when participant systems have been connected to the **System Coupling** system and their physics setups are complete (that is, their **Setup** cells are in an up-to-date state).

Note:

You can also access this option on the **Project Schematic** by right-clicking the **System Coupling** system's **Setup** cell.

Exports the files needed for a user-interface run to a specified working directory. For more information, see Exporting a System Coupling Setup (p. 273).

Coupling-Specific Properties

Use the **Setup** node's coupling-specific properties to define analysis details. To access properties, select any of the sub-nodes defined under **Setup**. Any properties associated with the selected node are displayed below under **Properties**, where you can edit the properties that are enabled.

Coupling properties are defined under the following nodes:

Analysis Settings

Participants

Data Transfers

Execution Control

Analysis Settings

Defines the basic behavior of the analysis. When **Analysis Settings** is selected, the following properties are shown below under **Properties**:

Analysis Type

Initialization Controls

Duration Controls

Step Controls

Analysis Type

Property defining the type of the analysis.

Possible values:

- **General**

  Used when all participants are executing a steady-state or static solution.

- **Transient**

  Used when any participant is executing a transient solution (includes mixed steady-transient analyses).

Initialization Controls

Coupling Initialization

Property controlling how the analysis is initialized:

Possible values:

- **Program Controlled**

  For initial runs (that is, not restart runs), the initial time and step are each set to **0**.

  For restart runs, the initial time and step are set to the values obtained from the most recent valid restart point.

- **Restart Points** (indicated by Step and Time)

  A coupled simulation can have multiple restart points when Intermediate Restart Data Output (p. 257) is selected for either all coupling steps or for a set of coupling step intervals. The next coupled analysis will be started based on the restart point you select.

  For more information, see Restarting a Coupled Analysis in Workbench (p. 268).

> **Important:**
>
> Program-controlled or explicitly specified restart points affect only the coupling step and/or time used to restart System Coupling. Appropriate restart points must also be specified for the co-simulation participants that are part of the coupled analysis.

Duration Controls

The duration of the analysis is defined by the following properties.

Duration Defined By

Property controlling how the duration of the analysis is defined:

Possible values:

- **End Time**

  Available when the **Analysis Type** is **Transient**

  System Coupling executes coupling steps until the end time specified by the **End Time** property is reached. In a transient analysis, each coupling step is a time step (with the

time interval specified by the step size). Note that the final coupling step size is reduced automatically, if needed, so that the specified end time is respected.

Note:

Some participant systems, such as Mechanical, require that the end time specified in their setup is respected. When a coupled analysis involves one or more participants that require their end time to be respected, then the maximum allowable end time for the coupled analysis is the minimum of the end times reported by such participants. In this case, a validation error will be reported if the coupled analysis' specified end time is greater than the minimum identified.

Other participant systems, such as Fluent, can run past the end time specified. These participant systems have no effect on the allowable end time of the coupled analysis.

- **Number of Steps**

  Available when the **Analysis Type** is **General**.

  System Coupling executes coupling steps until the number of steps specified by the **Number of Steps** property have been executed.

End Time [s]

Available when **Duration Defined by** is set to **End Time**. Property defining the duration of the analysis.

Required. A value must be specified before a *Solve* operation is attempted.

Default value is **None**. Accepts real values greater than **0.0**.

If units are not specified, seconds **[s]** are used.

Number of Steps

Property defining the number of coupling steps in the analysis. Available when **Duration Defined by** is set to **Number of Steps**.

Step Controls

The duration of the coupled analysis is broken into a sequence of ***coupling steps***. Data transfers between the coupled solvers occur at the beginning of each coupling iteration within a coupling step. Coupling steps are always indexed. During the analysis, each new coupling step is started when:

- The coupling analysis duration has not been reached, and
- Either the maximum number of coupling iterations has been reached or the coupling step has converged.

The coupling steps of the analysis are defined by the following properties:

Step Size

Available when the coupling is defined in terms of time (a transient analysis). Property defining the time interval associated with each coupling step.

Default value is **None**. Accepts real values greater than **0.0**.

If units are not specified, seconds **[s]** are used.

Note:

The final coupling step size is reduced automatically, if needed, so that the specified end time is respected. This reduction does not occur if the analysis duration is set by the **Number of Steps**.

The coupling step size is fixed for the duration of the coupled analysis but can be changed when restarting the analysis.

Minimum Iterations

Property defining the fewest number of coupling iterations that can be executed per coupling step (must be at least 1).

The specified minimum number of coupling iterations is executed even if all measures of convergence are realized in fewer iterations.

Maximum Iterations

Property defining the greatest number of coupling iterations that can be executed per coupling step.

The specified maximum number of coupling iterations may not be executed if the analysis converges before the maximum iteration step is reached.

Participants

Defines the participants involved in the analysis.

Under the **Participants** node, a read-only sub-node is defined for each participant connected to **System Coupling** system's **Setup** cell on the **Project Schematic**. The sub-node takes the participant name, as presented in the **Project Schematic**

Under a given participant node, a Regions (p. 250) node is defined.

Regions

Defines the regions for the participant. Contains a collection of participant regions from/to which data can be transferred.

A ***region*** is most often a point, line, surface, or volume that is part (or all) of the participants geometry or topology. Note, however, that equations or probe (monitored) values may also be considered as point regions.

Note:

System Coupling in Workbench requires participants to use 3D meshes, with data transfer regions consisting of element faces from a 3D mesh. System Coupling, when run from one of its user interfaces, supports both 2D and 3D meshes.

When a given region is selected, the following properties are shown below under **Properties**.

Topology

Property indicating the topology of the region. Currently, the only available value is **Surface**.

Note:

Volume regions are available for GUI or CLI-based co-simulations with Maxwell as a participant.

Input Output

Property indicating the variable (such as force, length, or temperature) transferred to/from the region.

Physical Type

Property indicating the quantity transferred to/from the region.

Data Transfers

Defines the data transfers to be performed in the analysis.

System Coupling in Workbench defines a *data transfer* (p. 239) as one source region and one target region, with the transfer of one variable type in one direction between two participants. For a two-way data transfer on one region, two individual data transfers are defined.

Note:

For data transfer creation rules applicable to both System Coupling and System Coupling in Workbench, see Rules for the Creation of Data Transfers (p. 281).

Under the **Data Transfers** node, a sub-node is defined for each data transfer created. When a given data transfer is selected, the following properties are shown below, under **Properties**:

Source/Target

Data Transfer Control

Data Transfer Context Options

Source/Target

Data transfer source and target are defined by the following properties:

Source/Target

Defines the **Source** and **Target** for the data transfer. Both are defined by the following properties:

Participant

Defines the participant associated with the data transfer.

Region

Defines the region to/from which data are transferred.

Variable

Defines the variable to be transferred.

When setting up data transfers, follow a top-down approach when defining **Source** and **Target**. Select in this order:

1. **Source Participant**
2. **Source Region**
3. **Source Variable**
4. **Target Participant**
5. **Target Region**
6. **Target Variable**

Data Transfer Control

Data transfer behavior is defined by the following properties:

Transfer At

Controls when the data transfer is executed by the solver. Only available option:

- **Start of Iteration**

  Transfer data at the start of every coupling iteration within a coupling step.

Under-Relaxation Factor

Factor multiplying the current data transfer values when under-relaxing them against the previous values. This is overridden with unity in the first coupling iteration of every coupling step only when the **Analysis Type** is **Transient**.

For more detailed information, see Under-Relaxation Algorithm (p. 302).

Note:

- Ramping and under-relaxation are independent operations. Ramping is applied before under-relaxation.
- When under-relaxation is used, there is no guarantee that the full value from the source side of the data transfer is applied to the target by the end of the coupling step.

RMS Convergence Target

Target value used when evaluating convergence of the data transfer within a coupling iteration. The default value is **1e-2**. The convergence target is RMS-based. For information regarding how this target is applied, see Evaluating Convergence of Data Transfers (p. 278).

Ramping

Determines the behavior of the ramping algorithm for specified quantity.

Ramping is used by System Coupling to slow the application of the source-side value on the target side of the interface.

Possible values:

- **None**

  No ramping is applied. The full data transfer value is applied to the target side of the interface for all coupling iterations. (default value)

- **Linear to Minimum Iterations**

  Within each coupling step, the ramping factor is used to linearly increase the change in the data transfer value applied to the target side of the interface.

For more information on the ramping algorithm, see Ramping Algorithm (p. 300).

Note:

System Coupling's ramping interacts with the ramping behaviors of the participant systems. To understand the full ramping behavior, verify ramping settings to see if your participant system is ramping loads received from System Coupling. For ramping behavior in Mechanical, see System Coupling Related Settings in Mechanical in the *Mechanical User's Guide*.

Data Transfer Context Options

After you connect a participant system's **Setup** cell to the System Coupling **Setup** cell in the **Project Schematic** and refresh the **Setup** cell, context options for creating and manipulating data transfers are available.

To access these context options, right-click the **Data Transfers** node or any specific data transfer defined underneath it.

Available options:

Create Data Transfer

Use to create single and multiple data transfers in any of the following ways:

- **Create an uninitialized data transfer:**

  Right-click **Data Transfers** and select **Create Data Transfer**. This creates a new data transfer without any source or target properties defined. You can change the data transfer definition later, in **Properties** pane.

- **Create data transfers for two regions from different participants:**

  Multi-select two regions from different participants, right-click one of them and select **Create Data Transfer**. This creates multiple data transfers that vary based on the following criteria:

  – Whether the two regions have the same topology

  – Whether the input variable from one region has the same properties (such as the physical type) as the output variable from the other region

- **Create data transfers for a single region:**

  Right-click a single region and select **Create Data Transfer**. This creates data transfers for each variable associated with the region. If the variable is an output variable, then the source participant, source region, and source variable are defined for the new data transfer. If the variable is an input variable, then the target participant, target region, and target variable are defined for the new data transfer.

- **Create a data transfer for a single variable:**

  Select a region, then right-click a variable under **Properties** pane and select **Create Data Transfer**. This creates a new data transfer. If the selected variable is an output variable, then the source participant, source region, and source variable are defined for the new data transfer. If the selected variable is an input variable, then the target participant, target region, and target variable is defined for the new data transfer.

Rename

Right-click a data transfer, select **Rename**, type in the new name, and click **Enter**. Alternatively, you can double-click a data transfer to rename it.

Duplicate Data Transfer

Select one or more data transfers, right-click one of them, and select **Duplicate**. This operation creates new data transfers with the same **Source**, **Target**, and **Data Transfer Control** properties. Note that you can change these properties as needed.

Delete

Select one or more data transfers, right-click one of them, and select **Delete**.

Suppress

Select one or more unsuppressed data transfers, right-click one of them, and select **Suppress**.

Unsuppress

Select one or more suppressed data transfers, right-click one of them, and select **Unsuppress**.

Note:

If the data transfer definition is not valid or the data are invalidated for any reason, the state of the node shows as *Attention Required* ().

Execution Control

Provides the following properties to control the execution of the coupled analysis.

Coupling Engine

Expert Settings

Intermediate Restart Data Output

Coupling Engine

The **Coupling Engine** property is available only for coupled analyses created using System Coupling 1.0, which has been removed as of the 2020 R2 release. When **Execution Control** is selected, the **Coupling Engine** property is shown under **Properties**.

The purpose of the **Coupling Engine** property is to allow you to set the analysis to run with the current release of System Coupling (2.0). The only value possible is **2.0**.

To continue working with a 1.0 simulation using the 2020 R2 release, you must set the **Coupling Engine** property to **2.0** for each **System Coupling** system on the **Project Schematic**.

When this value is selected and saved, the current version of the System Coupling engine will be used to run the analysis. This entails the following changes:

- All data previously generated for the case are cleared.
- Once switched, the case cannot be reverted back to compatibility with System Coupling 1.0.
- After this System Coupling session is ended, the **Coupling Engine** property will no longer be available.

Note:

If you wish to continue working with the case as it is (that is, without switching it to use the 2.0 coupling engine, as described above), then you must use a pre-2020

R2 release of System Coupling. If you do not switch to the 2.0 coupling engine and attempt to proceed using the case with the 2020 R2 release, updating the **Solution** cell of a **System Coupling** system will raise an error.

Expert Settings

Expert settings provide you with advanced controls for some System Coupling properties. For more information, see:

Working with Expert Settings in Workbench

Expert Settings Available in Workbench

Working with Expert Settings in Workbench

To enable an expert setting:

1. In System Coupling **Outline**, expand **Execution Control**.
2. Right-click **Expert Settings** and select **Add Property**.
3. Select the expert setting to be enabled.

The selected setting is shown under **Properties** with a value of **True** (enabled).

To edit an enabled expert setting:

1. In **Outline**, select **Expert Settings**.
2. Under **Properties**, select a different value for the setting.

The setting value is changed.

To disable an expert setting:

1. In **Outline**, right-click **Expert Settings** and select **Remove Property**.
2. Select the expert setting to be disabled.

The setting is removed from **Properties**.

Expert Settings Available in Workbench

The following expert settings are available when using System Coupling in Workbench:

ExportInterfaceDataPerStep

When enabled, generates interface/data transfer diagnostics files for each coupling step. At the end of the final iteration of each coupling step, System Coupling generates two `.axdt` files for each data transfer, with one file for each participant, to its working directory.

For a given step, these files include the mesh for all nodes (mapped and unmapped) and the results for data transfer variables corresponding to the final coupling iteration in that step.

Disabled by default.

GlobalStabilization

When enabled, Quasi-Newton stabilization is used for the coupled analysis. System Coupling in Workbench supports only default global stabilization settings. For more information, see Quasi-Newton Stabilization Algorithm (p. 303).

Tip:

To access the full functionality of the Quasi-Newton algorithm, you may export the coupling setup from Workbench so the analysis can be executed in System Coupling's GUI or CLI. For details, see Exporting a System Coupling Setup (p. 273) and Running an Exported System Coupling Setup (p. 190).

WriteDiagnosticsDictionary

When enabled, generates mapping weight and interpolation diagnostic files. System Coupling writes the following `.json` files to the **Solution Information** pane and its working directory:

- A single `weightsDiagnostics_step<#>.json` file is generated during the first iteration of the first step of the coupled analysis. For a given interface, this file contains weights diagnostics for the mapping process of that interface.
- An `interpolationDiagnostics_step<#>_iteration<#>.json` file is generated for every iteration in the coupled analysis. For a given step, these files contain full interpolation diagnostics. Contents include mesh and results — specifically, the mapped/unmapped mesh locations and data transfer variable(s) corresponding to the final iteration of the step.

Enabled by default.

Source_InitValue_HeatTransferCoef

Sets the initial source value for a Heat Transfer Coefficient variable. When enabled, specified initial value is used as the raw variable value provided by the source participant to initialize the simulation (that is, for the first iteration of the first step of the simulation).

If an actual raw value has been provided by the source participant, the specified initial value overrides it.

Accepts real values **0.0** $\leq$ **x** $\leq$ **1e+30**. Default value is **0.0**.

Intermediate Restart Data Output

When **Intermediate Restart Data Output** is selected, the following properties allow you to define details about the restart data generated during the execution of the solution:

Output Frequency

Specify the frequency with which restart files are generated.

Possible values:

- **None**

  No intermediate restart output files are generated using this option. (default value)

- **All Steps**

  Restart output files are generated at the end of each coupling step.

- **At Step Interval**

  Restart output files are generated at the end of the coupling steps corresponding to the interval specified by the **Step Interval** property.

Step Interval

Available when **Output Frequency** is set to **At Step Interval**. Specifies the interval at which restart files are generated. For example, with a value of 3, System Coupling generates a restart point at the end of every third coupling step.

Accepts integer values greater than or equal to **1**. Default value is **1**.

Note:

Depending on the participant, the restart data may or may not be the same as the results data. Writing of results data for postprocessing should be set from within the participant **Setup** cell.

Important:

During execution of the coupled analysis, co-simulation participants are automatically requested to generate intermediate restart data at the same frequency as System Coupling. Note that this feature only affects the frequency at which data are generated. The content of data is determined by the participant.

Running a Coupled Analysis in Workbench

This section describes the properties and controls used to run a coupled analysis using System Coupling in Workbench.

- All the properties that appear in the **Outline of Schematic** and **Properties** panes under the **Solution** node.
- Context options for the **Solution** node and child nodes defined underneath it.

The following topics are covered:

Starting the Solution

Monitoring Solution Progress

Solution Validation and State

Solution Node Context Options

Note:

For information on executing an exported coupling setup in one of System Coupling's user interfaces, see Running an Exported System Coupling Setup (p. 190).

Starting the Solution

Once the coupled analysis setup is complete, start the solution. In the **System Coupling** tab's **Outline**, right-click **Solution** and select **Update**.

Alternatively, you can start the solution from the **Project Schematic** by right-clicking the **System Coupling** system's **Solution** cell and selecting **Update**.

Monitoring Solution Progress

As the solution proceeds, you can monitor its progress as follows:

Viewing Transcript Output

Viewing System Coupling Charts

Viewing Transcript Output

Solution information is automatically generated for output of System Coupling and the coupling participants. Figure 32: Solution Information node (p. 260) shows an example of the **Outline of Schematic**'s **Solution Information** node. Select one of the solution information sources underneath it to view its output in the **Solution Information** pane.

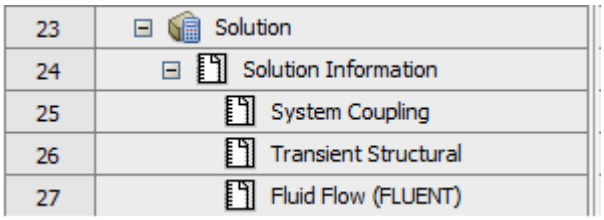
Note:

The default behavior of the **Solution Information** pane is to always show the latest information in the Log file. Each time information is added, the file automatically scrolls to the end. However, if you move the vertical scroll bar away from the bottom, the pane will not scroll to the end when information is added until you move the scroll bar back to the end.

There are also some keyboard short-cuts that are available when operating in this pane:

- **Page Up** scrolls up one page.
- **Page Down** scrolls down one page.
- **Ctrl**+**Home** jumps to the top of the Log.
- **Ctrl**+**End** jumps to the bottom of the Log.

Figure 32: Solution Information node



For details on the solution information shown when you select **System Coupling**, see Transcript and Log File (p. 332). For details on the solution information shown for a given coupling participant, use the links in Supported Coupling Participants for System Coupling in Workbench (p. 238) to access its product documentation.

Note:

When **Beta Options** are enabled in Ansys Workbench, you can also select your own text files to monitor in the **Solution Information** pane. To do this, right-click the **Solution Information** entry in the **Outline** pane, choose **Create Solution Information (Beta)** and select the desired file.

Viewing System Coupling Charts

The **System Coupling Chart Monitor** displays the coupling chart selected in the **System Coupling** tab's **Outline of Schematic**. System Coupling charts plot the data produced by System Coupling during its execution. They are updated dynamically during the analysis, so you can use them to monitor convergence as the solution progresses.

Note:

Currently, System Coupling in Workbench does not plot participant quantities. However, charting of participant quantities is available in System Coupling's interfaces. For more information, see Reviewing Convergence Diagnostics Charting Output (p. 197).

For information on monitoring fluid participant quantities in a CFD participant product, see one of the following sections

- Monitoring and Reporting Solution Data
- CFX-Solver Manager Monitors Menu
- CFD-Post Monitor Menu.

In the **Outline of Schematic**, you can use the context options (p. 264) for the **Chart Monitors** node to create multiple charts, and then use the context options for the individual **Chart** nodes to select the variables (p. 263) to be plotted.

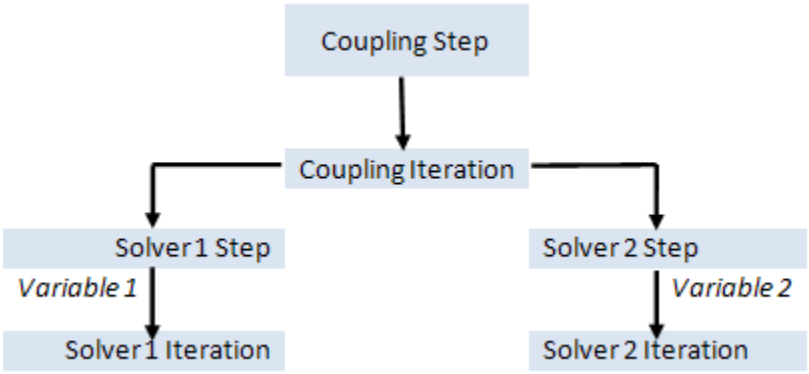
For information on working with the charts generated by System Coupling in Workbench, see:

Availability of Chartable Data

Availability of Chartable Data

Multiple levels of **X-axis** data are available for plotting. The higher (coarser) levels of data are called *parent levels*, and the lower (finer) levels of data are called *child levels*. Any data present in a child level is also available to its parent level. For example, Figure 33: Hierarchical availability of plotting data (p. 261) shows a flowchart illustrating the availability of plotting data for an sample coupled analysis run.

Figure 33: Hierarchical availability of plotting data



In this coupling run:

- **Coupling Step** is the parent level for
  - **Coupling Iteration**, which is the parent level for
    - → **Solver 1 Step** and **Solver 1 Iteration**, and
    - → **Solver 2 Step** and **Solver 2 Iteration**.

  The two variables, **Variable 1** and **Variable 2**, with each being available for plotting at their respective solver iterations and, by extension, all of the iteration's parent levels:

  - **Variable 1** is available at the **Solver 1 Iteration** level, so is also available at parent levels **Solver 1 Step**, **Coupling Iteration**, **Coupling Step**.
  - **Variable 2** is available at the **Solver 2 Iteration** level, so is also available at parent levels **Solver 2 Step**, **Coupling Iteration**, **Coupling Step**.

Chart Properties

When a **Chart** object is selected in the **System Coupling** tab's **Outline of Schematic**, its properties are shown in **Properties of Convergence Chart**:

Axis X

Contains the following property:

Quantity

Level at which the X data for the variables is plotted. Possible values, which are determined by the type of analysis being performed, are:

- **Coupling Step**
- **Coupling Iteration**
- **Coupling Time** (transient analyses only)

Axis Y

Contains the following property:

Title

Title of the Y axis for the chart.

Axis X, Axis Y

Properties supported for both axes:

Scale

Scale of the axis.

Possible values:

- **Linear Common**
- **Log** (Log base 10)
- **Natural Log**

Automatic Range

Defines whether automatic scaling or range maximum and minimum are applied to the axis.

- When selected, automatic scaling is used.
- When not selected, the range specified by the range minimum and maximum properties is used.

Range Minimum

Available when the **Automatic Range** property is not selected. Minimum of the range of values for the axis.

Range Maximum

Available when the **Automatic Range** property is not selected. Maximum of the range of values for the axis.

Chart Variables

The **System Coupling Chart Monitor** plots data produced during the coupled solution process. It can plot the following variables:

- the RMS change in data transfer values
- diagnostic values (sum and weighted average) taken from the nodal data associated with data transfers

For more information on System Coupling chart variables, see:

Adding a Chart Variable

Chart Variable Properties

Adding a Chart Variable

You can add a variable to a chart by using its **Chart** node's **Add Variable** context option (p. 264). Variables are organized by coupling participant and data transfer, as shown in Figure 34: Selection of Convergence Chart variable (p. 263).

Figure 34: Selection of Convergence Chart variable

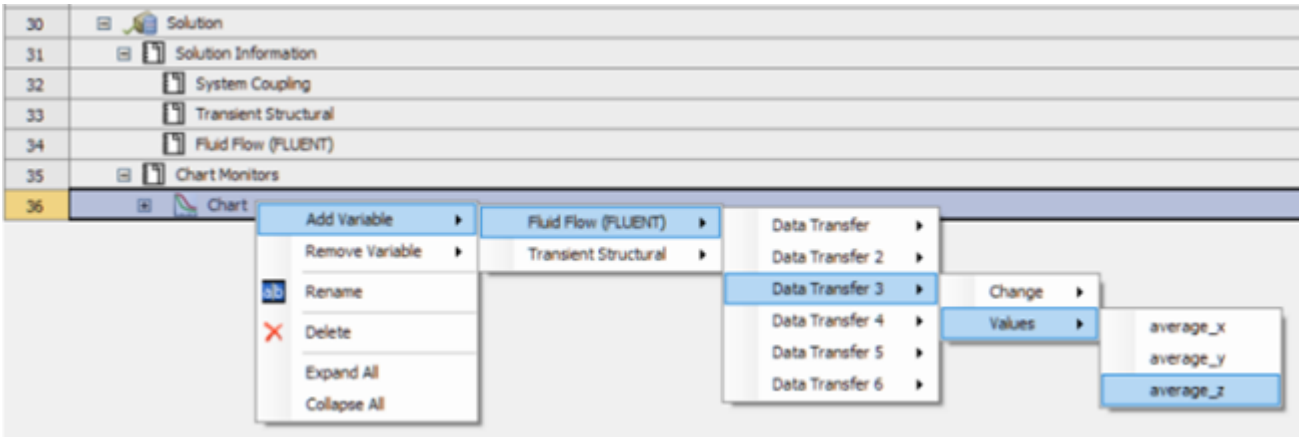


Chart Variable Properties

The variables to be plotted for a chart are listed under its **Chart** node in the **Outline of Schematic**. When a chart variable is selected, its properties are shown in **Properties of Chart Variable**.

Refinement Level

Specifies the level refinement for X axis variable data. Variable data plotted at the X axis level can be further refined to any of the X axis' child levels. For example, if the X axis **Quantity** is defined at the **Coupling Step** level, then the X data values for its variables can be refined to the **Coupling Iteration** level.

In this case, the intermediate values available at the coupling iteration level between consecutive coupling steps are distributed equally between the coupling steps — that is, if coupling step 2 has three coupling iterations, then the data points are plotted at 1.33, 1.66, and 2.

Style

Contains the following properties that define the appearance of the chart:

Color

Defines the line color of the chart variable in a plot.

Line Width

Defines the width (in pixels) of the line for this chart variable.

Symbol Size

Defines the size of a symbol in (pixels) when a symbol is drawn for this variable.

Chart Context Options

Context options are available for working with System Coupling charts. In the **System Coupling** tab's **Outline of Schematic**, you can access these options by right-clicking the **Chart Monitor** node or any chart or chart variable nodes defined underneath it.

Create Chart

Available for the **Chart Monitors** node.

Creates a new System Coupling chart without any variables defined. By default, the new chart's X axis **Quantity** is set to the **Coupling Iteration** level.

Add Variable

Available for the **Chart** nodes.

Variables can be added to a chart after the solution has started or completed. Right-click an existing chart, select **Add Variable**, and then select the variable to be added.

By default, the new variable's **Refinement Level** is set to the same level as the X axis **Quantity** property. If data for the new variable is not available at that level, then the X axis **Quantity** property and the variable's **Refinement Level** properties are both set to **Coupling Iteration**.

Remove Variable

Available for the **Chart** nodes.

Right-click an existing chart, select **Remove Variable**, and then select the variable to be removed.

Delete

Available for the **Chart** and chart variable nodes. Deletes the selected item from the chart.

Rename

Available for the **Chart** and chart variable nodes. Allows you to rename the selected item.

Note:

When the solution is started, a default chart is added if one is not already present. The default variables added correspond to the RMS change in data on the target side of all data transfers. For example:

If **Data Transfer** and **Data Transfer 2** are defined with target participants equal to **Transient Structural** and **Fluid Flow**, respectively, then the chart variables **Transient Structural: Data Transfer: Change: RMS** and **Fluid Flow: Data Transfer 2: Change: RMS** are added to the default chart.

If you add/delete variables to the default chart, then new variables are not added by default on consecutive runs.

Chart Monitor Controls

You can manipulate the display of a chart using the ***Zoom***, ***Pan***, and ***Fit*** operations:

- *Zoom* by using the mouse wheel or **Shift**+**middle mouse button**
- *Box zoom* by using the **right mouse button**
- *Pan* by using **Ctrl**+**middle mouse button**
- *Fit* by using the **F** key.

Saving a Chart as a Graphic

To save a System Coupling chart as a graphic:

1. Right-click the background of the chart and select **Save Image As**.

   A dialog that displays, showing a small image of the chart.

2. Select the **Size** (resolution) to be used when saving the chart.
3. Click the ellipsis () and navigate to the folder where you want to save the file.
4. Enter a filename.
5. Select a graphic file type (`.png` or `.bmp`).

6. Click **Save** and then click **OK** to save the file to the location and at the resolution you have selected.

Note:

When **Quantity** is set to **Coupling Time**, the first data point may be generated at 0 instead of the time the first coupling step finishes. The second and all subsequent data points are then generated at the correct times, according to the **Time Step Size** defined.

Solution Validation and State

The state of the **Solution** node in the **System Coupling** tab's **Outline of Schematic** corresponds to the state of **System Coupling** system's **Solution** cell on the **Project Schematic**.

In turn, both of these are tied to the states of the **Solution** cells for all co-simulation participant systems on the **Project Schematic**. The **Solution** node and all coupled **Solution** cells will have the same state, which will reflect the least complete state of all coupled cells.

Solution Node Context Options

Context options are available for working with System Coupling solutions. In the **System Coupling** tab's **Outline of Schematic**, you can access these options by right-clicking the **Solution** node or any of the child nodes defined underneath it.

- **From the Solution node, you can:**
  - **Update** the solution;
  - **Continue Calculation** if the solution was interrupted;
  - **Refresh** the solution;
  - **Clear Generated Data**; and
  - **Reset** the solution.

  These commands are the same as those available on System Coupling's **Solution** cell in the **Project Schematic**.
- **From the Chart Monitor node, you can:**
  - Right-click **Chart Monitor** and select **Create Chart** to create a new System Coupling chart.
  - Right-click a **Chart** node and select **Add Variable** or **Remove Variable** to change the variables plotted on the chart.
  - Right-click a chart variable and select **Rename** or **Delete**.
- **Display Validation Failure**:

Select this option to display error messages when System Coupling's **Solution** items are found to be incorrect due to validation problems.

Note:

If a coupled analysis is interrupted before reaching the specified coupling duration, then the **Solution** cells stay in an *Update Required* state once execution stops because the calculation must be continued to complete the analysis. You can refresh and/or update downstream **Results** cells to review the results generated up to the point at which the analysis was interrupted.

For more information on typical cell states, see Understanding Cell States in the *Workbench User's Guide*.

Common Tasks for System Coupling in Workbench

For information on tasks that are commonly used in workflows for System Coupling in Workbench, see:

Stopping a System Coupling Analysis in Workbench

Restarting a Coupled Analysis in Workbench

Stopping a System Coupling Analysis in Workbench

To stop a coupled analysis in Workbench:

1. In the Workbench **Progress** pane, click **Stop** ().
2. On the dialog that displays, select from the following options:
   - **Interrupt**: Performs a clean shutdown. The analysis stops once the current coupling step is completed.
   - **Abort**: Stops the analysis immediately. All available generated data are discarded.
   - **Cancel**: Continues with the current run.

When you click **Interrupt** or **Abort**, a Shutdown (p. 329) file is automatically generated in System Coupling's working directory and tells both System Coupling and participant services to shut down.

You can also create this file manually and place it in the working directory, as described in Shutdown File (p. 329). In this case, however, you must end any participant processes yourself.

Once the analysis has been stopped, the Shutdown file is automatically removed from System Coupling's working directory.

Restarting a Coupled Analysis in Workbench

You can restart a completed or interrupted coupled analysis in Workbench, either from the final result or from an intermediate result. When restarting a coupled analysis Workbench, you must select the necessary participant restart files and start participants manually.

Note:

Mechanical **Steady-State Thermal Transient Thermal** analysis system do not support restarts.

For more information, see:

Generating Restart Files in Workbench

Restarting a Coupled Analysis in Workbench

Recovering from a Workbench Crash

For information on restarts from one of System Coupling's user interfaces, see Restarting a Coupled Analysis (p. 142).

Generating Restart Files in Workbench

Restarts of a System Coupling analysis require corresponding restart points to exist in System Coupling and in each of the solvers participating in the analysis.

Restart points are generated at the end of a coupling step. To generate restart files for a coupled analysis:

1. Before starting the initial run of the analysis, ensure that all coupling participants are set up to save (or retain) the corresponding restart points during the analysis. For information on a given participant, see Supported Coupling Participants for System Coupling in Workbench (p. 238) for a list of supported systems and references to their corresponding documentation regarding restarts.

2. Set up the **System Coupling** system to control the creation of restart points at certain intervals during the execution of the coupled analysis.

   a. On the **Project Schematic**, double-click System Coupling's **Setup** cell.

      The **System Coupling** tab opens.

3. In the **Outline**, select **System Coupling** > **Setup** > **Execution Control** > **Intermediate Restart Data Output**.

4. In **Properties** pane, set **Output Frequency** to one of the options described in Intermediate Restart Data Output (p. 257).

Restarting a Coupled Analysis in Workbench

Once the execution of the Workbench coupled analysis is finished or interrupted, or if the solution fails, you can restart from any of the saved restart points. You must select the same restart point for the **System Coupling** system and all coupling participants.

Note:

For coupled analyses set up in Workbench, restarts are not supported if the default participant names (**Solution**, **Solution 1**, and so on) have been changed.

To restart the coupled analysis run:

1. For each participant connected to the **System Coupling** system, specify a restart point in the participant application. Make sure that these restart points correspond to the restart point you will choose in System Coupling.

   For instructions on selecting a restart point for a given participant, see Supported Coupling Participants for System Coupling in Workbench (p. 238) and follow the link to its documentation.

2. Make any necessary changes to analysis settings for participants.

   For instructions on changing settings for a given participant, see Supported Coupling Participants for System Coupling in Workbench (p. 238) and follow the link to its documentation.

   Note:

   Changes saved prior to the restart (either in System Coupling or in any participant except Mechanical) are written to the Settings (p. 351) file and included in the restarted run.

   ---

3. Make any necessary changes to the **System Coupling** system:

   a. Double-click System Coupling's **Setup** or **Solution** cell to open the **System Coupling** tab.

   b. Change the required System Coupling settings. Setup changes commonly include changes to a combination of the following:

      - Coupling analysis type
      - Coupling initialization and duration settings
      - Coupling step size
      - Minimum and maximum number of coupling iterations per coupling step
      - Data transfer convergence target and under-relaxation factors

4. Select the restart point for the **System Coupling** system:

a. On the **Project Schematic**, open the **System Coupling** tab by double-clicking System Coupling's **Setup** or **Solution** cell.

b. In the **Outline** under **Analysis Settings > Properties**, set **Coupling Initialization** to the restart point corresponding to the restart point you selected in the participant systems.

5. In the **Outline**, right-click **Solution** and select **Update** to restart the solution.

Details for the restarted run are appended to the end of the existing `scLog.scl` file. System Coupling goes to the end of the specified coupling step and continues the solution with the next step.

Recovering from a Workbench Crash

Workbench or one of its component systems may crash in such a way that restart files are available, but not recognized or populated in the Workbench project. In this case, you can recover your project and restart your analysis using the steps below.

The usual project directory (`<ProjectName>_files`) contains the latest System Coupling results and restart points (these solvers use the live project instead of running in a temporary directory).

Note that the **.backup** directory contains the original version of any files which have been modified since the last save. These files are useful to recover the last saved state, but they are not useful for restarting your analysis.

To recover the project so you can restart from a specified point:

1. Launch Workbench and open the project.

2. The project will be locked because it was not shut down cleanly. When asked what you want to do with the project, click **Unlock**.

3. When asked if you want to recover the last saved state before opening, click **No**.

   The state shown on the **Project Schematic** shows that the solution had not started, but you will find that backup files are available. Note, however, that your Workbench project does not know about these files.

4. Populate the restart data from the participant systems connected to the **System Coupling** system, ensuring that these points correspond to the restart point specified in System Coupling.

   For information on how to do this for a given participant, see Supported Coupling Participants for System Coupling in Workbench (p. 238) and follow the links to its documentation.

5. Recover System Coupling's restart points:

   a. On the **Project Schematic**, right-click System Coupling's **Setup** cell and select **Update**.

   b. Double-click the **Setup** cell to open the **System Coupling** tab.

   c. In the **Outline**, select **Analysis Settings**, then in **Properties of Analysis Settings**, right-click **Coupling Initialization** and select **Read Restart Points**.

      The restart points will now be available in System Coupling as usual.

d. In **Properties**, for **Coupling Initialization**, pick a restart point that corresponds to the restart point you selected in the participant systems.

e. Right-click **Solution** and select **Update**.

Your restarted coupled analysis will begin to solve.

Advanced Tasks for System Coupling in Workbench

For information on tasks that may be used in advanced workflows for System Coupling in Workbench, see:

Requesting Debugging and Diagnostic Output

System Coupling can generate diagnostics and debugging output to help you troubleshoot issues with coupled analyses. For more information on requested this output from System Coupling in Workbench, see:

Requesting Debug Log Files

System Coupling can generate debug log files to record transactions between itself and its participant solvers. Log file content is controlled by the command-line options listed in Debug Logging Command-Line Options in the *System Coupling Settings and Commands Reference* documentation.

To request that debug log files are generated, perform the following steps:

1. On Workbench's **Project Schematic**, right-click the **System Coupling** system's **Solution** cell and select **Properties**.

   Corresponding properties are shown to the right in the **Properties of Schematic: Solution** pane. Under **General**, note the **Command Line Options** property.

2. Enable the generation of debug log files.

   In the **Command Line Options** property's **Values** field, type in the `-l` or `--logLevel` option, setting it to a value of **1** or higher.

3. Optionally, you can use the following additional arguments to specify details for the log output:

   - Use the `--logDir` option to specify an output directory for the log files.

   Note:

   By default, the log files are written to System Coupling's working directory within the project's **dp0** directory (**dp0\SC\SC**).

- If the logging level is set to a value of ***2*** or higher, use the `--logMaxArrayElements` to specify the maximum number of data array elements that can be included in the file. (Default value is ***5***.)

When the **System Coupling** system's **Solution** cell is updated, debug logging is activated.

During the solution, System Coupling will write data to the following files, with "*`date`*" and "*`time`*" indicating when the run began:

`SyC_Log_Controller_<`*`date`*`>_<`*`time`*`>.txt`

Contains log information from the controller process.

`SyC_Log_CNode<`*`N`*`>_<`*`date`*`>_<`*`time`*`>.txt`

Contains log information from compute node "*`N`*"

Tip:

The log file for compute node `0` usually contains the most relevant information.

Example 51: Request that Level 1 debug log files are written to the default directory

Log files containing RPC calls will be written to System Coupling's working directory.

```
--logLevel=1
```

Example 52: Request that Level 2 debug log files are written to the specified directory

Log files containing RPC calls and diagnostics will be written to the **SyC/DebugOutput** directory within System Coupling's working directory.

```
> "%AWP_ROOT251%\SystemCoupling\bin\systemcoupling" -G --logLevel=2 --logDir=SyC/DebugOutput
```

Example 53: Request that Level 3 debug log files are written to the default directory

Log files containing RPC calls/diagnostics and a maximum of three heavyweight data-array elements will be written to System Coupling's working directory.

```
$ "$AWP_ROOT251/SystemCoupling/bin/systemcoupling" -l3 --logMaxArrayElement=3
```

Exporting a System Coupling Setup

You can create a coupled analysis in Workbench and export its setup for execution in one of System Coupling's user interfaces. Because all the steps are automated, this is the preferred method of making a Workbench-based analysis available in System Coupling's GUI or CLI.

Note:

The import of `.sci` files via System Coupling's **File > Import SCI File** GUI option or `ImportSystemCouplingInputFile()` command is still supported, but not recommended.

The setup process is similar to that for an analysis to be run in Workbench, as described in Setting Up a Coupled Analysis in Workbench (p. 243) (though completion of the **System Coupling** system is optional). When the setup of participant systems (p. 243) is complete (including schematic connections between participant systems' **Setup** cells and the **System Coupling** system's **Setup** cell), you can export the files needed to execute the analysis in the GUI or CLI. The capabilities available for the coupled analysis are determined by the timing of the export within the Workbench workflow. The export operation automatically creates a directory structure and writes the necessary System Coupling and participant files to it.

For more information, see:

Steps to Create the System Coupling Setup

Files Needed for a User-Interface Run

Exporting Partial vs. Full Coupling Setups

Exporting the System Coupling Setup

Steps to Create the System Coupling Setup

To create the System Coupling setup in Workbench, perform the following steps:

1. **Complete the setup for each coupling participant system.**

   Set up the physics for each participant in the same way as for a Workbench run, as described in Completing the Coupling Setup (p. 245).

   Note:

   Ensure that the paths to all required scripts and executables are added to the PATH environment variable so that participating applications can be launched from the GUI or CLI.

   ---

2. **Optionally, complete the setup for the System Coupling system.**

   Whether you should perform this step depends on the type of coupling setup file you want to use for the execution of the analysis. For details, see Files Needed for a User-Interface Run (p. 274).

Whether you should perform this step before the export depends on the capabilities required for the coupled analysis. For details, see Exporting Partial vs. Full Coupling Setups (p. 274).

3. **Export the System Coupling setup.**

   Export the files necessary for the user-interface run, as described in Exporting the System Coupling Setup (p. 275).

Files Needed for a User-Interface Run

When the coupling setup is exported, a coupled analysis working directory and subdirectory structure is created. The following required files are automatically written:

Solver Input Files

An input file for each participant solver is written to a participant working directory created inside the coupled analysis working directory.

System Coupling Settings File

Setup information is written to a System Coupling Settings file (p. 351) in the **SyC** folder created in the coupled analysis working directory.

The contents of the Settings file, which reflects the state of the data model at the time of the export, determines the coupling capabilities that are available for the analysis. For more information, see Exporting Partial vs. Full Coupling Setups (p. 274).

Exporting Partial vs. Full Coupling Setups

Depending on the capabilities required for the analysis, you can export either a partial or a full coupling setup, as follows:

Partial Setup:

To export a partial coupling setup, perform the export *before* the coupling setup is completed in Workbench. When a partial coupling setup is loaded into System Coupling's GUI or CLI:

- Additional coupled analysis setup steps must be completed in the System Coupling user interface. Participant information is loaded, but **Coupling Control**, **Coupling Interface**, and **Data Transfer** settings must be specified.
- All capabilities supported by the user-interface context (p. 23) are available for the coupled analysis. A partial setup provides access to the broadest range of capabilities and gives the most control over the coupled analysis configuration.

Full Setup:

To export a full coupling setup, perform the export *after* the coupling setup is completed in Workbench. When a full coupling setup is loaded into System Coupling's GUI or CLI:

- No additional setup steps are required. Once the setup is loaded into the System Coupling user interface, the solution is ready for execution.

- Only the capabilities supported by the Workbench context (p. 23) are available for the coupled analysis.

Note:

When exporting a full coupling setup, restarts are not supported if the default participant names (that is, **Solution**, **Solution 1**, and so on) have been changed.

Exporting the System Coupling Setup

To export the System Coupling setup, perform the following steps:

1. On Workbench's **Project Schematic**, right-click the **System Coupling** system's **Setup** cell and select **Export System Coupling Setup**.

   The **Browse for Folder** dialog opens.

2. Navigate to the location that will be the working directory for the coupled analysis. This should be a folder outside the Workbench project directory structure.

3. Select the working directory (or click **Make New Folder** to create one).

4. Click **OK**.

This folder is now designated as System Coupling's working directory for the coupled analysis. A folder structure similar to the one described in Recommended Directory Structure (p. 116) is created inside it. When the export is completed, it contains the following items:

SyC directory

Directory created by System Coupling for the storage of its own output files. It contains the following file:

- **Settings file** (`Settings.h5`) (p. 351)

  System Coupling file populated by content from the Workbench setup. This file is used to load the coupled analysis into one of System Coupling's user interfaces.

Export Setup Log file

Export Setup Log file (`exportSetup.log`) (p. 352) containing summary information for all the coupled analysis systems, folders, and files involved in the export.

Participant working directories

A working directory for the storage of input and output files is created for each participant. Participant working directories are named according to the following convention:

```
<ParticipantID>_<#>
```

where:

"*ParticipantID*" is the participant system's **Component ID** property, accessed by right-clicking the **Project Schematic** and selecting **Properties**

"*#*" is an index to ensure uniqueness of the system identifier. For example, the working directories created for two Fluent participants would be named **FFF** and **FFF-1**

System Coupling input files

System Coupling input files — either participant `.scp` files or an `.sci` file, depending on whether a partial or full coupling setup was exported — are written to the working directory. The sole purpose of these files, however, is to populate the `Settings.h5` file.

Generating Participant Solver Input and Coupling Setup Files On Demand

Exporting the System Coupling setup (p. 273) is the recommended method of making a Workbench-based case available for a System Coupling user-interface run. However, it is possible to request that a participant generates its solver input and `.scp` files on demand.

This process for generating on-demand files varies according to participant type (that is, whether the participant solver is CFX, Fluent, or Mechanical). For instructions on how to do this for a given participant, see Supported Coupling Participants for System Coupling in Workbench (p. 238) and follow the link to the participant's coupling-specific documentation.

Because you are not using the automated workflow described in Exporting the System Coupling Setup (p. 275), you must perform all the export steps manually, as follows:

1. Create System Coupling's working directory, including sub-folders to serve as participant working directories, as described in Setting Up a Coupled Analysis Directory Structure (p. 116).
2. On the **Project Schematic**, verify the following details:
   - The participants systems' **Setup** cells are connected to the **System Coupling** system's **Setup** cell.
   - The participant systems' setups are complete — that is, all cells upstream of the **Solution** cells should be in an *Up-to-Date* state ().
3. In each coupling participant product, request the generation of solver input and `.scp` files, ensuring that the files are written to the participant's coupled analysis working directory.
4. Ensure that the paths to all required scripts and executables are added to the PATH environment variable so that participating applications can be launched from System Coupling GUI or CLI.

Coupled Analysis Management

One of the main roles of System Coupling is to manage the coupled analysis. During execution, System Coupling performs a variety of one-way and two-way data transfers between coupling participants.

One-way or two-way coupling is defined relative to a coupling participant. If the participant only sends data to System Coupling or only receives data from System Coupling, then it is engaged in one-way coupling. If the same participant sends data to System Coupling (for example, heat flow) and receives data from System Coupling (for example, temperature), then it is engaged in two-way coupling.

When multiple participants execute their individual parts of a coupled analysis together, it is called a ***co-simulation*** . In co-simulation, each participant can engage in both one-way and two-way data transfers as either a source or target and perform their solutions simultaneously.

For details on the various aspects of the coupling management process, see:

Inter-Process Communication

Convergence Management

Evaluating Convergence of Data Transfers

Inter-Process Communication

System Coupling and participants, which are often highly optimized physics solvers, are executed as independent computational processes, and this introduces the need for Inter-Process Communication (IPC). The IPC is managed internally by System Coupling via Remote Procedure Calls (RPC).

Solvers interact with System Coupling using the System Coupling Participant (SCP) library. The library interfaces are documented on the Ansys Devloper Portal.

Convergence Management

By default, System Coupling's Log file (p. 332) reports Root Mean Square (RMS) convergence for data transfers for both the source and target side of the transfer. Convergence of the coupling step is evaluated at the end of each coupling iteration. Coupling step convergence requires that:

- the target side RMS values have reached the convergence criteria that you specified in the input to the System Coupling setup, and
- that the minimum number of coupling iterations that you specified are met.

If the coupling step is not yet converged, then a new coupling iteration is started. If the coupling step has converged, then a new coupling step is started if the coupling duration has not yet been reached.

Evaluating Convergence of Data Transfers

To evaluate the convergence of data transfers, each iteration is measured against the previous iteration. The change in all the data transfer values between these two successive iterations is reduced to a normalized value. When two successive iterations produce a normalized value that is under the convergence (you can change the convergence target default value of ***1e-2***), the data transferred has converged.

Two global (that is, over all locations) measures of convergence are evaluated and reported during execution of the coupled analysis. These include the maximum and Root Mean Square (RMS) of the normalized change in data transfer values. The RMS is the default measure used to determine convergence. The measure can be changed to the maximum of the normalized value through the System Coupling Input file.

The RMS value is evaluated as:

$$\text{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\hat{\Delta}_i^2} \tag{1}$$

where $\hat{\Delta}$ is the normalized change in the data transfer value between successive iterations within/across a given coupling step, and is measured as:

$$\hat{\Delta} = \frac{1}{0.5\times((\max|\varphi| - \min|\varphi|) + |\overline{\varphi}|_{\text{ave}})}\Delta \tag{2}$$

where φ is the data transfer value.

In Equation 2 (p. 278), the denominator, or normalization factor, is evaluated differently in the transient and general coupling analyses. In the transient coupling case, the normalization factor equals the average of the range and mean of the magnitude of data transfer values over all locations for the current iteration. In the general coupling case, it equals the average of the range and mean of the magnitude of data transfer values over all locations for all iterations in the entire analysis. This normalization factor is a representative scale for the data transfer values and ensures that division by zero (due either to zero range or zero mean) is avoided.

In Equation 2 (p. 278), Δ is the un-normalized change between successive iterations, and is expressed as:

$$\Delta = \frac{1}{\omega}\left(\varphi - \varphi^{o}\right) \tag{3}$$

where φ and φ^{o} correspond to the current and the previous iterations respectively, and ω is the under-relaxation factor applied in forming the final value applied during the current iteration. In the first coupling iteration of every coupling step, ω is assumed to be unity. If the variable is a vector and/or complex, then φ and φ^{o} represent the variable magnitude.

When there is no change in data transfer values, the default for RMS/MAX is ***1.0e-014***.

Note:

The global data transfer convergence measures are set to unity in the first coupling iteration of the first coupling step during an initial run. After a restart, if a data transfer involving a

new variable is defined or if the region is remeshed, these measures are set to unity in the first coupling iteration of the first coupling step.

Although monotonic convergence to the specified target values is ideal, oscillatory convergence and/or divergence (that is, constant or increasing convergence measures) may also occur.

For best practices, see Evaluating Convergence and Data Transfer Accuracy (p. 397).

System Coupling Data Transfers

During a coupled analysis, one of System Coupling's core functions is to manage data transfers between the coupling participants. A ***data transfer*** is the transfer of a single quantity type in one direction between regions on the source participant and target participant.

System Coupling currently supports transfers of any scalar or vector quantity and any real or complex data type.

The source participant region must be able to send the quantity and the target participant region must be able to receive the quantity type being transferred.

For example, consider a coupled one-way thermal analysis in which Fluent and Mechanical are the participants. In this co-simulation, the fluids region is the source for the transfer of Temperature and the structural region is the target for the transfer of that Temperature. This could be extended to a two-way thermal analysis by introducing transfer of Heat Rate from the structural region to the fluids region.

Force and displacement transfers are typical for fluid-structure interaction problems, where a load to the structure is transferred from a fluid solver and the deformations to the fluid are transferred from the structural solver.

For information that is applicable to data transfers in general, see the following sections:

Rules for the Creation of Data Transfers

System Coupling's Data Transfer Process

System Coupling's Mapping Capabilities

Supplemental Processing Algorithms

Context-specific information on data transfers is also available:

- For information about data transfers in System Coupling's user interfaces, see Coupling Interfaces and Data Transfers (p. 56).
- For information about data transfers in System Coupling in Workbench, see Data Transfers for System Coupling in Workbench (p. 239).

Rules for the Creation of Data Transfers

The following quantity-specific limitations apply:

- On a given interface, if a transfer of Force is defined, then a transfer of Incremental Displacement cannot also be defined in the same direction, and vice versa.
- A single region cannot be the source of both Temperature and Heat Rate, nor the target of both Temperature and Heat Rate.

- Regarding the Heat Transfer Coefficient (HTC) and Convection Reference Temperature (Tref) data quantities:
  - For a given source/target pair of regions, if a transfer of HTC is defined, then a transfer of Tref must also be defined, and vice versa.
  - A transfer of HTC+Tref is equivalent to a transfer of Heat Rate. If a transfer of HTC+Tref is defined for then a transfer of Heat Rate cannot be defined, and vice versa.

Note:

System Coupling and System Coupling in Workbench each define and handle data transfers slightly differently. For context-specific details, see Rules for Adding Data Transfers (p. 57) or Data Transfers for System Coupling in Workbench (p. 239).

System Coupling's Data Transfer Process

System Coupling's data transfer process can be divided into two components, each of which executes its own set of algorithms:

Mapping (p. 282)

Mapping is the process of using data on a source mesh to calculate data on a target mesh. System Coupling uses various mapping algorithms perform the following tasks:

- Associate source and target locations
- Generate data for target locations

Supplemental Processing (p. 299)

Supplemental processing may be applied to data before and/or after mapping. System Coupling uses various supplemental processing algorithms to perform the following tasks:

- Reconstruct data on mesh locations
- Apply ramping and under-relaxation factors
- Apply unit conversions as needed
- Apply Quasi-Newton stabilization

System Coupling's Mapping Capabilities

In System Coupling, all data transfers are accomplished via ***mapping***, which is the process of using the data source to calculate data on the target. Source and target may be mesh or point cloud. If it's a mesh, the data can be located on mesh nodes or on mesh elements.

System Coupling performs two types of mapping, ***profile-preserving*** and ***conservative***.

The mapping process consists of two distinct phases:

1. Intersecting source and target to calculate mapping weights
2. Interpolating data from source to target

For more detailed information on mapping, see:

Mapping Types
Mapping Topologies
Region Discretization Types
Mapping Process
Associating Source and Target Locations
Generating Data for Target Locations

For best practices, see Improving Mapping Quality (p. 394).

Mapping Types

System Coupling performs two types of mapping, each with its own objectives: ***profile-preserving (p. 283)*** and ***conservative (p. 284)***.

Profile-Preserving Mapping

- The goal of profile-preserving mapping is to minimize the difference between the profile, or distribution of data, on the source and target mesh locations.
- Profile-preserving mapping is used to transfer ***intensive variables*** from interface source to target regions.
- Profile preservation is enforced to the extent possible, based on the least-resolved topology from the source/target region combination.
- Not all target locations receive data originating from the interface source. For target locations that are not mapped (that is, do not overlap with source locations), target data are generated based on the values of other mapped nodes or centroids. For details, see Generating Data for Target Locations (p. 289).
- Supports the following quantity types associated with intensive variables:

  Scalar quantities:

  - Temperature
  - Heat Transfer Coefficient
  - Convection Reference Temperature
  - Electrical Conductivity

  Vector quantities:

  - Incremental Displacement

  - Mode Shape

Conservative Mapping

- The goal of conservative mapping is to minimize the difference between the sum of the data (locally and globally) on the source and target mesh locations.
- Conservative mapping is used to transfer ***extensive variables*** from interface source to target regions.
- Not all target locations receive data originating from the interface source. Specifically, target locations that do not overlap with source locations are assigned a value of zero.
- Supports the following quantity types associated with extensive variables:

  Scalar quantities:

  - Heat Rate

  Vector quantities:

  - Force

In all cases, data from the source-side of the interface is used to calculate data on the target-side of the interface at either element or node locations. If algorithms require data on locations not provided or consumed by either side of the interface, then it is reconstructed by a supplemental algorithm, as described in Data Reconstruction Algorithms (p. 299). Diagnostics are provided for all mapping types to aid in assessing the quality and accuracy of target data.

Mapping Topologies

System Coupling can create mappings between participant regions of the following topologies:

- **Surface**
- **Volume**

Mapping can occur between participant regions of like topology, and in some cases, between participant regions of unlike topologies (for example, surface-to-volume). Support for different combinations of topologies in a given transfer varies according to the coupling context, as shown in the table below.

Table 27: Supported topology combinations for co-simulation mapping

Topology		GUI	CLI	WB
Source	Target			
Surface	Surface	✓	✓	✓
Volume	Volume	✓	✓	
Surface	Volume	✓	✓	
Volume	Surface	✓	✓	

Region Discretization Types

Regions have an internal discretization type which may be set using RegionDiscretizationType. For co-simulation, the region discretization type is set to the default value of ***`Mesh Region`***.

When set to ***`Point Cloud Region`***, System Coupling supports standalone mapping and co-simulation of intensive variables between regions of like topology. These include:

- 2D Surface Point Cloud to/from 2D Surface Point Cloud
- 2D Surface Point Cloud to/from 2D Surface Mesh
- 3D Surface Point Cloud to/from 3D Surface Point Cloud
- 3D Surface Point Cloud to/from 3D Surface Mesh
- 3D Volume Point Cloud to/from 3D Volume Point Cloud
- 3D Volume Point Cloud to/from 3D Volume Mesh

Note:

Region discretization types are not supported for System Coupling in Workbench.

Mapping Process

System Coupling's mapping process consists of two phases: the creation of associations between source and target meshes (p. 286), and the generation of data on target mesh locations (p. 289). Mapping is performed whenever System Coupling receives new participant meshes, which may be when the analysis is initialized and/or when participant meshes are resubmitted, as described below:

Mapping at Coupled Analysis Initialization

In most coupled analyses, coupling participants serve their initial meshes only once, at the beginning of a run (either the initial run or a restarted run), when the coupled analysis is initialized.

The resulting mapping is used for all subsequent coupling steps and iterations in the analysis. Mapping diagnostics for all mapped interfaces are written to System Coupling's Transcript, in the **Mapping Summary** (p. 342) section at the beginning of the first coupling step.

Note:

When a coupled analysis is restarted, if it contains at least one AEDT participant, mapping is not performed during initialization.

Mapping at the Beginning of a Coupling Step

When an AEDT participant updates its mesh and resubmits it to System Coupling, this triggers other coupling participants to resubmit their meshes. Mapping occurs at the beginning of a coupling step each time participant meshes change, as follows:

- **If initialization mapping did not occur**, then the initial mesh import and generation of mapping weights occur at the beginning of the first iteration of the first coupling step.
- **If the mesh has changed** (either from the initial mesh or from the mesh used in a previous step), then participant meshes are reimported and the mapping weights are recalculated at the start of the first iteration of the current coupling step. The resulting mapping is used for all subsequent coupling steps and iterations until the mesh changes again.
- **If the mesh has not changed**, then the current mapping is used for all subsequent coupling steps and iterations until the mesh changes again.

Associating Source and Target Locations

To begin the mapping process, System Coupling creates associations between locations on the source mesh and locations on the target mesh, essentially placing them in the same simulation space. The association process begins with a ***Binary Space Partitioning (BSP)*** search to quickly identify the required set of candidate mesh locations to be used in the generation of mapping weights. For more information, see Binary Space Partitioning Algorithm (p. 287).

When a source location overlaps a target location and an association is created, the target location is said to have been successfully "mapped." When no overlap exists and no association can be made, the target location is said to have been left "unmapped."

There are four methods used to associate source and target locations: ***Projection (p. 286)***, ***Coincident (p. 286)***, ***Extrusion (p. 287)***, and ***Search Radius (p. ?)***. The method used is determined by the topologies and region discretization type of the participant regions involved in the mapping, as follows:

Projection

- Used for **surface-to-surface** (2D-to-2D) mapping when the region discretization type is set to ***`Mesh Region`***.
- Mesh from one side of the interface is projected (or scattered) onto the other side of the interface, as follows:
  - For **profile-preserving mapping**, the target mesh is projected onto the source mesh. For an illustration, see Shape Functions (p. 292).
  - For **conservative mapping**, the source mesh is projected onto the target mesh. For an illustration, see Intersection Algorithm for Surface-to-Surface and Volume-to-Volume Mapping (p. 296).

Coincident

- Used for **volume-to-volume** (3D-to-3D) mapping when the region discretization type is set to ***`Mesh Region`***.
- Source mesh is mapped directly to the target locations.
- For an illustration of **profile-preserving mapping**, see Radial Basis Functions (p. 294).
- For an illustration of **conservative mapping**, see Intersection Algorithm for Surface-to-Surface and Volume-to-Volume Mapping (p. 296).

Extrusion

- Used for **volume-to-surface** (3D-to-2D) and **surface-to-volume** (2D-to-3D) mapping.
- Source mesh is extruded onto the target mesh.
- For an illustration of **profile-preserving mapping**, see Element-Weighted Averages (p. 295).
- For an illustration of **conservative mapping**, see Intersection Algorithm for Surface-to-Volume and Volume-to-Surface Mapping (p. 297).

Note:

In these cases, the surface must be within one diameter of the volume mesh for mapping to occur. For best results, ensure that the orientation of the surface and its location with respect to the volume are consistent with your problem setup.

Search Radius

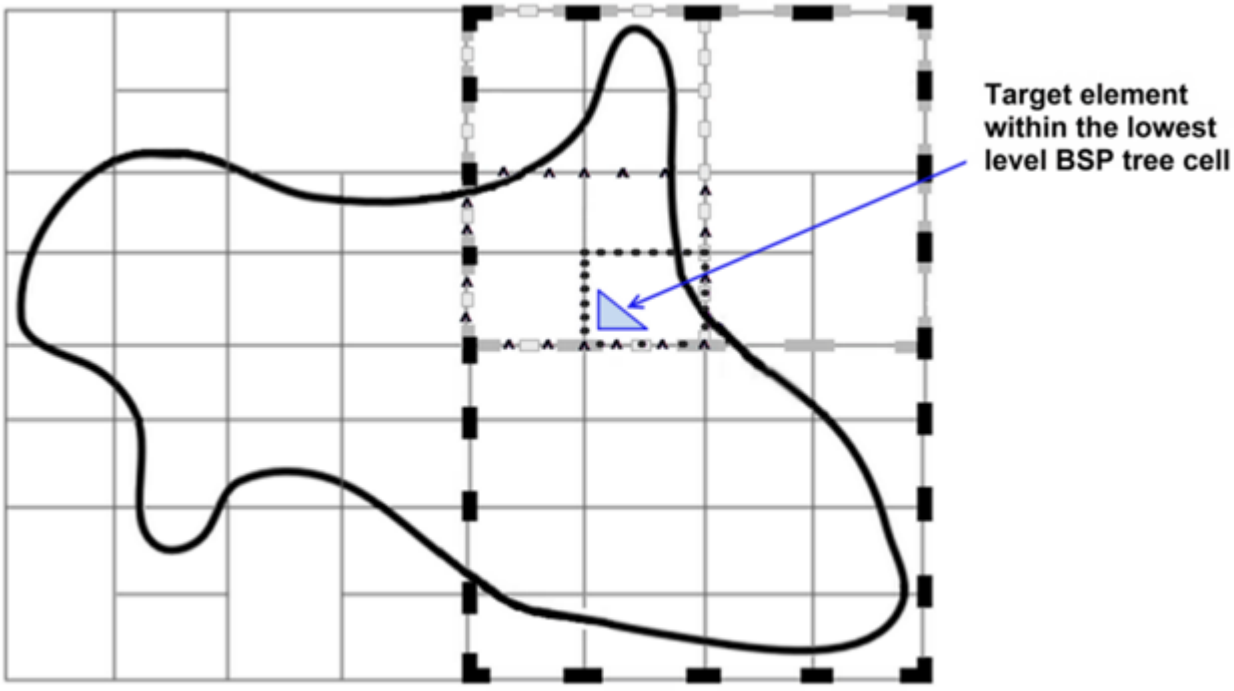
- Used for point cloud-to-point cloud mapping.
- Nearby source points are identified for each target location.
- For an illustration of **profile-preserving mapping**, see Radial Basis Functions (p. 294).
- Conservative mapping is not defined in this case.

Binary Space Partitioning Algorithm

A ***Binary Space Partitioning (BSP)*** tree algorithm is used to quickly identify the required set of candidate mesh locations to use for generating mapping weights.

The BSP algorithm fits a uniform structured grid of two cells to the source region(s). Each cell that contains more than a predetermined number of source mesh locations is recursively subdivided into two new cells in a direction that is orthogonal to the preceding division. This continues to form a hierarchical tree structure, as shown in the figure below, until either the maximum number of levels in the hierarchy is reached or the predetermined number of source mesh locations per tree cell is reached. BSP tree cells at this level are referred to as "leaf cells."

Figure 35: Example of a BSP tree mesh on a 2D target domain, which extends naturally to 3D (source mesh locations not shown for clarity)



Each target mesh location is ultimately associated with one or more leaf cells. Efficiency is realized by associating the target mesh locations with a cell at each level within the tree hierarchy, starting with the highest (largest) cells first.

From here, the association process varies, as follows:

Profile-Preserving Mapping

- The association goes from **target to source**.
- Once a target node is associated with a leaf cell(s), it is associated with the most suitable source element and/or nodes.

Conservative Mapping

- The association goes from ***source to target***.
- Once the source element is associated with a leaf cell(s):
  - For transfers to **surface target regions**, the source element is associated with intersecting target elements. Care is taken to appropriately consider the relative orientation and normal distance between associated source and target elements.
  - For transfers to **volume target regions**, the source element is associated with intersecting target elements.

Generating Data for Target Locations

After System Coupling has associated source and target locations, it generates data on the mapped target location, and when applicable, the unmapped target locations.

When data is transferred to the target location, its fidelity is limited by the least-resolved side of the interface. For example, if the target mesh is significantly coarser than the source mesh, then fine-scale features of the source solution may be missed in mapping. Similarly, if the target mesh is significantly finer than the source mesh, then the target data may exhibit pathologies due to the coarseness of the source mesh (that is, piece-wise constant or linear variation of source data may be apparent over groups of elements on the target side).

For information on the methods used to data on target locations, see:

Generating Data on Overlapping Locations

Generating Data on Non-Overlapping Locations

Mapping Algorithms by Mapping Type, Topology, and Region Discretization Type

Profile-Preserving Mapping Algorithms

Conservative Mapping Algorithms

Generating Data on Overlapping Locations

System Coupling calculates mapping weights and then uses the weights to interpolate the source data onto the target.

The algorithm System Coupling uses is determined by the type of mapping being performed plus the topologies of involved participant regions. Available data-generation algorithms are listed below.

Profile-Preserving Mapping

Shape Functions (p. 292)

For profile-preserving mapping from surface source mesh regions to **surface target mesh regions**, mapping weights are generated using shape functions.

Note:

System Coupling's interpolation is always at least as accurate as the interpolation used by the source participant solver. For details, see the relevant participant product documentation.

Radial Basis Functions (p. 293)

For profile-preserving mapping from volume regions to **volume regions**, mapping weights are generated using radial basis functions.

For volume-to-volume mapping with source data on elements, an element-based RBF algorithm maps the data to overlapping locations.

Element-Weighted Averages (p. 294)

For profile-preserving mapping from volume mesh regions to **2D surface mesh regions**, weights are generated using an element-weighted average.

Conservative Mapping

Intersection (p. 296)

For conservative mapping with any combination of topologies, mapping weights are generated by intersecting the source and target mesh elements, followed by transferring of the data from source to target, based on the extent to which the source and target elements are overlapping.

Generating Data on Non-Overlapping Locations

System Coupling handles the lack of associated source data according to the type of mapping being performed plus the topologies of involved participant regions.

Note:

The concept of "non-overlapping" locations is not applicable for point clouds.

Conservative Mapping

Non-overlapping target elements are assigned a value of zero.

Profile-Preserving Mapping

Target data is generated on non-overlapping target locations based on the values of other mapped locations. The method used is determined by the topology of the target region, as follows:

- Copy or extrapolate target values from overlapping locations on the target side (depending on the value of the `UnmappedValueOption` setting)

  This method is used when the source and target both consist of surface mesh regions.

- Copy the value from the nearest source-side location

  This method is used for all other mapping algorithms.

Mapping Algorithms by Mapping Type, Topology, and Region Discretization Type

The table below shows the different association and target data generation algorithms for profile-preserving and conservative mapping between regions with different topology/region discretization type combinations.

Mapping Type	Topology		Region Discretization Type		Target Data Generation	
	Source	Target	Source	Target	On Overlapping Locations (Mapping Weights)	On Non-Overlapping Locations
Profile-Preserving	Volume	Surface	Mesh	Mesh	Element-Weighted Averages	Calculated from Overlapping Target Locations[1 (p. 291)]
			Mesh	Mesh	Shape Functions	Calculated from Overlapping Target Locations[1 (p. 291)]
	Surface	Surface	Point Cloud	Point Cloud	Radial Basis Functions	N/A
			Point Cloud	Mesh	Radial Basis Functions	N/A
			Mesh	Point Cloud	Shape Functions	N/A
			Mesh[2 (p. 291)]	Mesh	Radial Basis Functions	Nearest Source Point Value[3 (p. 292)]
	Volume	Volume	Point Cloud	Point Cloud	Radial Basis Functions	N/A
			Mesh	Point Cloud	Radial Basis Functions	Nearest Source Point Value[3 (p. 292)]
			Point Cloud	Mesh	Radial Basis Functions	Nearest Source Point Value[3 (p. 292)]
Conservative	Surface	Volume	Mesh	Mesh	Intersection	Zero value
	Surface	Surface	Mesh	Mesh	Intersection	Zero value
	Volume	Volume	Mesh	Mesh	Intersection	Zero value

1: For profile-preserving mapping onto a target surface, target values can be either copied or extrapolated from the nearest target-side node, depending on the value of the `UnmappedValue-Option` setting.

2: When source data are on element centroids and the element's neighborhood (surrounding elements that share a face with it), an element-based version of the Radial Basis Function algorithm is used.

Mapping Type	Topology		Region Discretization Type		Target Data Generation	
	Source	Target	Source	Target	On Overlapping Locations (Mapping Weights)	On Non-Overlapping Locations
3: May be either element centroid or nodal values.						

Profile-Preserving Mapping Algorithms

For data transfers of intensive variables, the following algorithms are used to generate data for mapped target locations:

Shape Functions

Radial Basis Functions

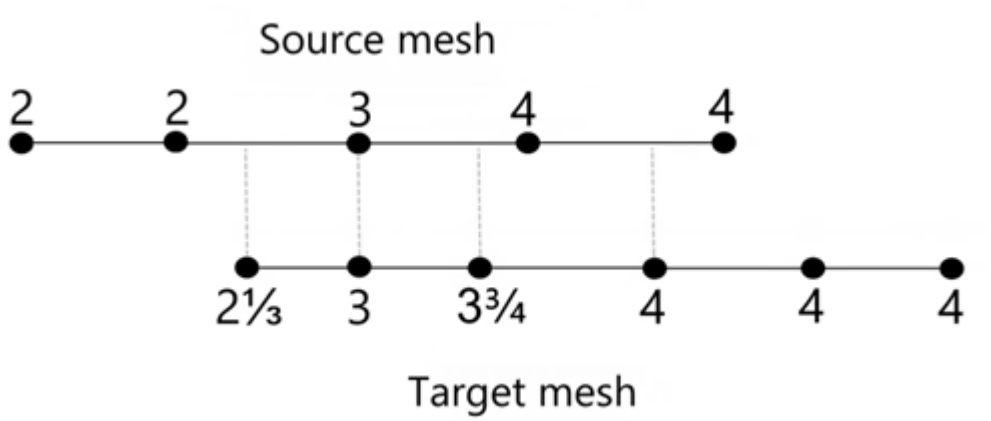
Element-Weighted Averages

Shape Functions

For transfers of intensive variables between surfaces, ***Shape Functions*** are used to generate target data for mapped locations. The type of shape functions selected are determined by the source mesh provided by the coupling participant. System Coupling uses linear shape functions for source meshes with linear elements, and quadratic shape functions for source meshes with quadratic elements.

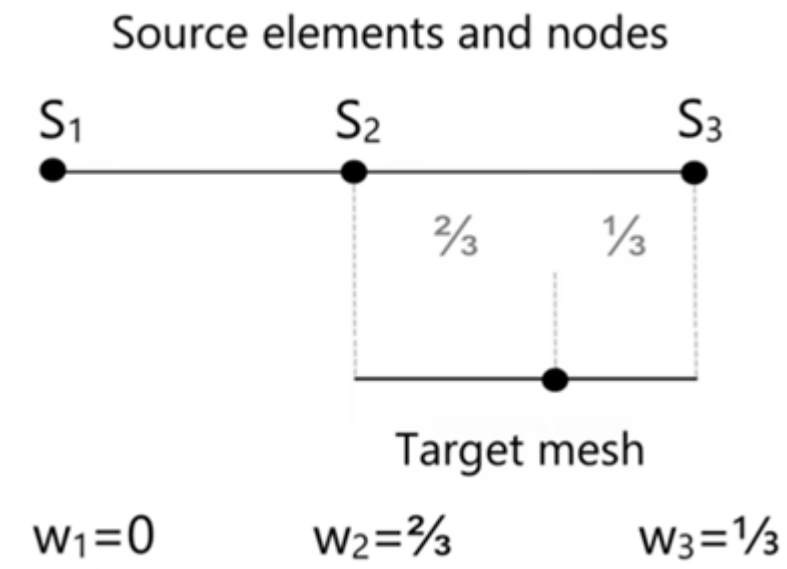
Quantities located on source mesh nodes are mapped to target mesh nodes, as shown in the figure below.

Figure 36: Example input and output for Shape Function mapping



Mapping weights are generated using finite element shape functions of the source element associated with each target node, as shown in the figure below.

Figure 37: Mapping weights generated for a target node, based upon its projection onto a source element via Shape Function mapping



Radial Basis Functions

For transfers of intensive variables between volumes, ***Radial Basis Functions*** (RBF) are used to generate target data for overlapping target locations.

The location of the source data determines the RBF algorithm used to map data to overlapping target mesh locations (either element centroids, mesh nodes, or point cloud nodes), as illustrated in the diagrams below.

- When source points are on **mesh nodes**:
  - A node-based RBF algorithm maps the data to overlapping target locations.
  - Non-overlapping target locations receive values from the nearest source point value.
- When source points are on element **centroids**:
  - An element-based RBF algorithm maps the data to overlapping target locations.
  - Non-overlapping target locations receive values from the centroid of the intersecting element.

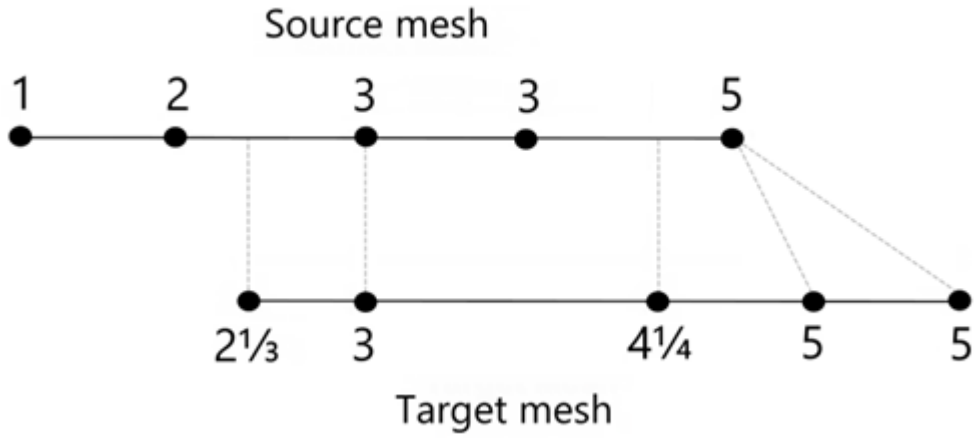
Generation of target data on non-overlapping locations is not applicable for point cloud-to-point cloud mapping.

Note:

- For volumes, the element-based RBF algorithm, which performs mapping directly on the element centroids, is used by default. A beta setting allows you to revert to the node-based RBF algorithm, which reconstructs elemental data on element nodes and then performs mapping on the reconstructed data.
- For surfaces, the element-based RBF algorithm is available as a beta feature.

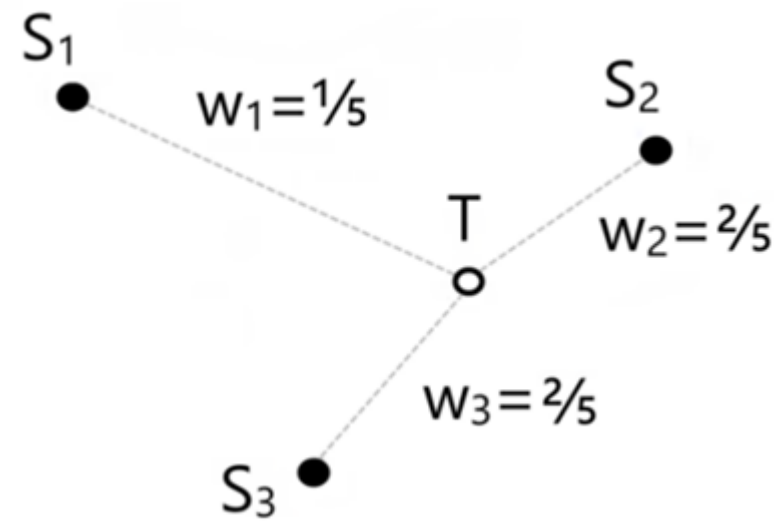
For details, see Activate System Coupling Hidden Features.

Figure 38: Example input and output for Radial Basis Function mapping



Mapping weights are generated using radial basis functions (RBFs) for the source nodes or centroids that define the associated source elements. Example weights are shown in the figure below.

Figure 39: Example mapping weights w_i generated for a target node (T), based on the distance to nodes with the associated source elements S_i via Radial Basis Function mapping

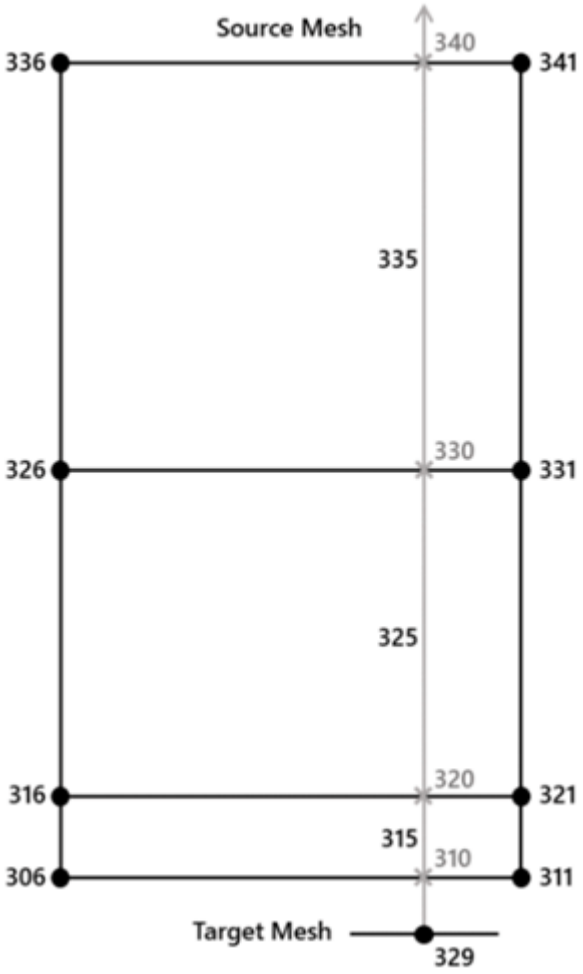


Element-Weighted Averages

For transfers of intensive variables from a volume to a surface, ***Element-Weighted Averages*** are used to generate target data for mapped locations.

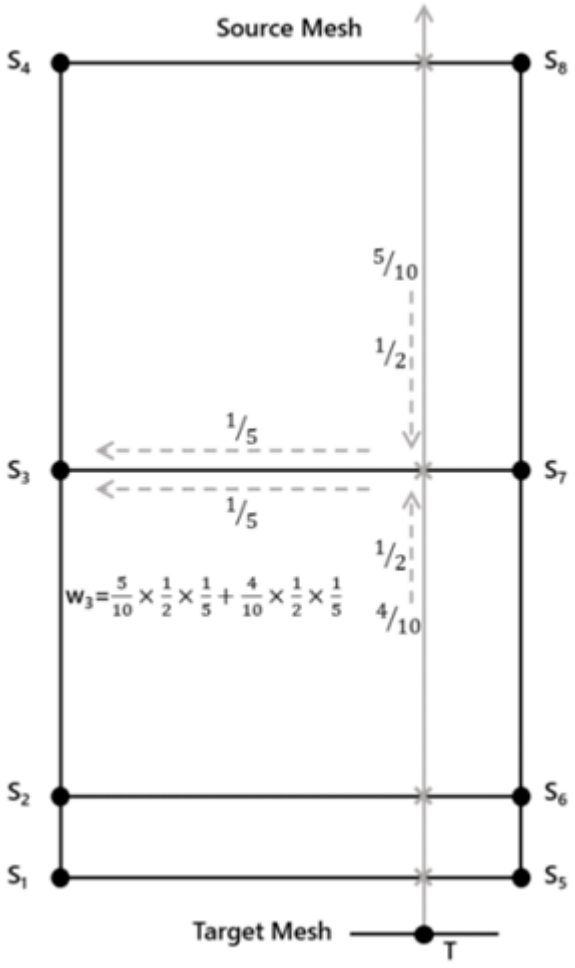
Quantities located on the source mesh nodes are mapped to target mesh nodes, as shown in the figure below.

Figure 40: Example input and output for Element-Weighted Average mapping



Intersection points are created by projecting the target (surface) nodes through the source (volume) elements. Mapping weights are generated by taking a length-weighted average of the line segment within each intersecting volume element. For an illustration, see the figure below.

Figure 41: Mapping weights generated for a target node (T), based upon its extrusion onto source elements via Element Weighted Average mapping. Representative weight calculation is shown for node 3; other weights are computed in a similar manner.



Conservative Mapping Algorithms

For data transfers of extensive variables, the Intersection algorithms are used to generate target data. There are two variations of this algorithm, and the variation used is determined by participant region topologies

Intersection Algorithm for Surface-to-Surface and Volume-to-Volume Mapping

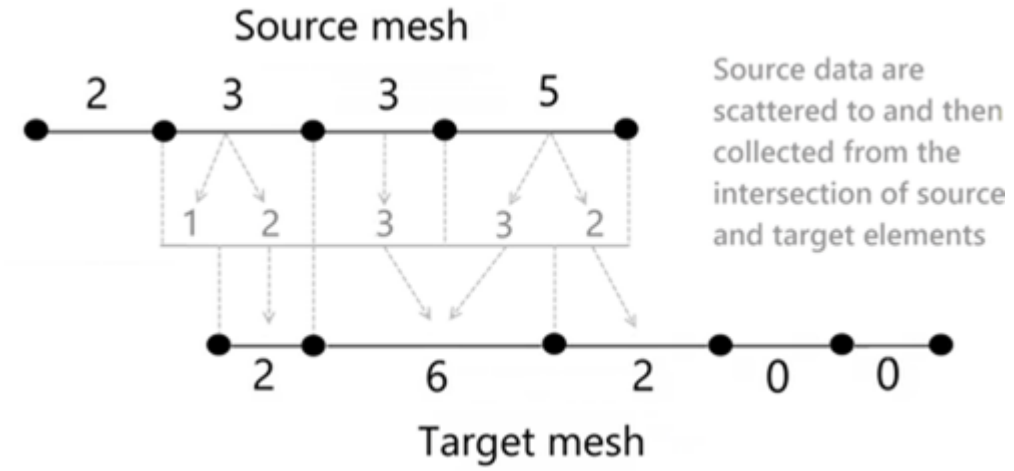
Intersection Algorithm for Surface-to-Volume and Volume-to-Surface Mapping

Intersection Algorithm for Surface-to-Surface and Volume-to-Volume Mapping

This version of the ***Intersection algorithm*** is used when extensive variables are transferred between two surfaces or between two volumes.

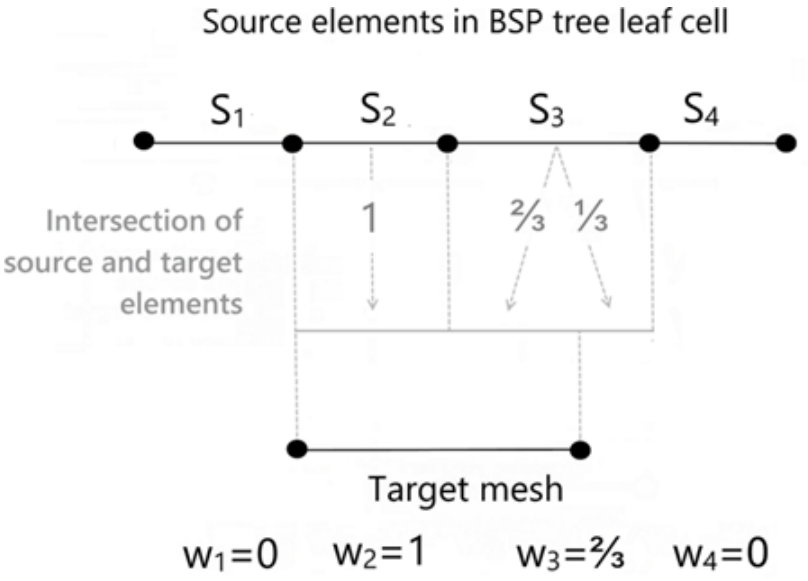
Data located on the source mesh elements are mapped to target mesh elements, as shown in the figure below.

Figure 42: Example input and output for conservative mapping between surfaces or between volumes



Mapping weights are generated based upon information extracted from intersecting the target element with the source elements. In particular, the size of each intersected region relative to the size of the respective source element is used, as shown in the figure below.

Figure 43: Mapping weights generated for a target element, based on its intersection with source elements between like topologies

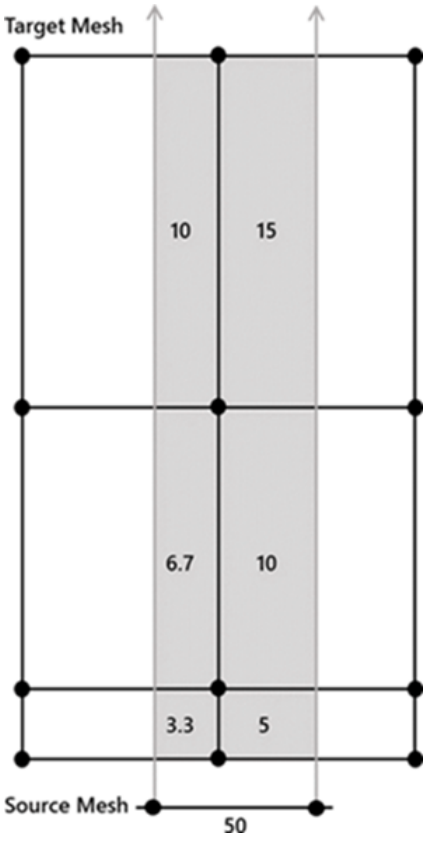


Intersection Algorithm for Surface-to-Volume and Volume-to-Surface Mapping

This version of the ***Intersection algorithm*** is used for unlike topologies (that is, applied when the **Extrusion** association method is used) is used when extensive variables are transferred from a surface to a volume.

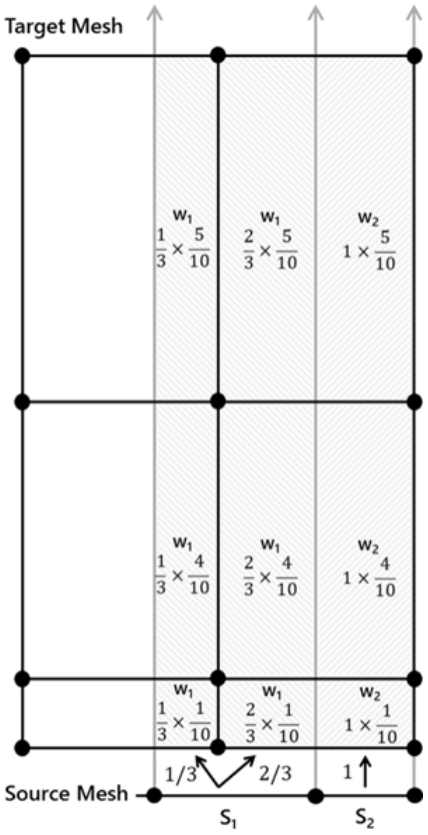
Data located on the source mesh elements are mapped to target mesh elements, as shown in the figure below.

Figure 44: Example input and output for conservative mapping for Intersection for surface-to-volume mapping. Circled values are the total for the elements.



When the target (surface) elements are extruded onto source (volume) elements, intersections are created between the resulting target (pseudo-volumetric) elements and the source (volume) elements. Mapping weights are generated by taking a volume-weighted average of the intersecting volume between the target and source elements. For an illustration, see the figure below.

Figure 45: Mapping weights generated for a target node, based upon its extrusion onto a target element via the Intersection algorithm for surface-to-volume mapping



Supplemental Processing Algorithms

Supplemental processing algorithms are used both before and after mapping to preprocess and postprocess the data being transferred. The following algorithms are available:

Data Reconstruction Algorithms

Ramping Algorithm

Under-Relaxation Algorithm

Quasi-Newton Stabilization Algorithm

Data Reconstruction Algorithms

In both the pre-interpolation are post-interpolation stages of the data transfer process, ***data reconstruction algorithms*** are used to create variable data on specific mesh locations (for example, nodes), given that data for this variable already exists at different mesh locations (for example, face centroids). These reconstruction algorithms are used when a source participant provides values at mesh locations different from those required by the mapper or when the mapper provides values at mesh locations different from those required by the target participant.

Conservative algorithms are used to reconstruct extensive variable data by one of the following methods:

Creating Nodal Data from Face/Element Centroid Data

Extensive variable values (such as forces or heat rates) may be available on element (faces or cells) centroids. If the values are required on nodes, the following steps are executed:

- For a given element, the contribution to the node value is calculated by dividing the value at the element by the number of nodes that define that element.
- These individual element contributions are accumulated at the node for each element adjacent to this node.

Intensive variable values (such as temperatures or incremental displacement) may be available on element (face or cell) centroids. If the values are required on nodes, they are calculated by taking an area- or volume-weighted average from all connected elements.

Creating Face/Element Data from Nodal Data

Extensive variable values (such as forces or heat rates) may be available on nodes. If the values are required on elements/faces, the following steps are executed:

- For a given node, the contribution to the element value is calculated by dividing the value at the node by the number of elements adjacent to that node.
- These individual nodal contributions are accumulated at each element for each node that defines that element.

Intensive variable values (such as temperatures or incremental displacements) may be available on nodes. If the values are required on elements/faces, they are calculated by taking a simple average of the nodal values on each element.

Ramping Algorithm

The ramping controlled by System Coupling is used to improve convergence of the overall analysis. It works by slowing the application of the source-side value on the target-side of the data transfer. For each data transfer location (node) where $i \leq N_{Min}$ is true, the following formula is applied:

$$\phi_{\text{Ramped}} = \phi_{\text{Reference}} + \frac{i}{N_{Min}} \left(\phi_{\text{Raw}} - \phi_{\text{Reference}} \right) \tag{4}$$

where

ϕ_{Ramped} is the ramped, target-side value.

$\phi_{\text{Reference}}$ is the reference target-side value, which for the first iteration of the first step in a coupling run (either an initial or a restarted run), the reference target-side value is interpolated by System Coupling. For the first coupling iteration of all subsequent steps, the reference target-side value is the final value from the last coupling step.

ϕ_{Raw} is the raw, target-side value obtained from interpolation.

i is the current coupling iteration number within the coupling step.

N_{Min} is the minimum number of coupling iterations per coupling step.

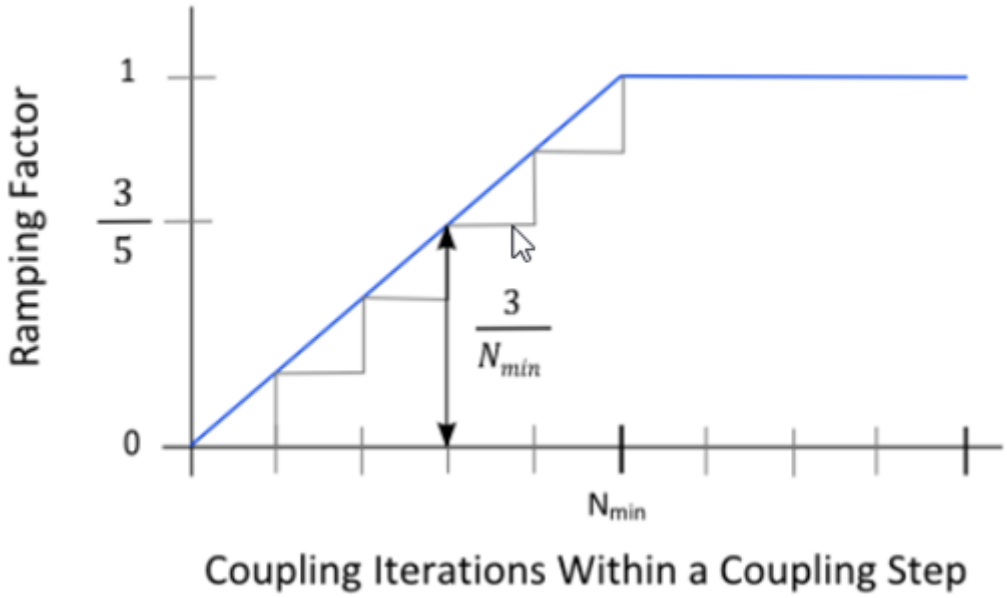
When ramping is enabled (that is, for System Coupling in Workbench, the **Ramping** setting is set to **Linear to Minimum Iterations**; or for System Coupling's GUI or CLI, the `RampingOption` setting is set to `Linear`), within each coupling step, the ramping factor is used to linearly increase the change in the data transfer value applied to the target side of the interface. The data transfer value is increased during each coupling iteration until the specified minimum number of coupling iterations, N_{Min}, is reached.

The ramping factor is applied to the change in the data transfer value from the last coupling iteration of the previous coupling step. If there is no change in this value from the last coupling step, the full data transfer value is applied to the target side of the interface for all coupling iterations of that coupling step.

During the i^{th} coupling iteration (for $i<N_{Min}$), the ramping factor equals i/N_{Min}. The full data transfer value is applied for all coupling iterations that are equal to or greater than the minimum number of coupling iterations. As N_{Min} is always reached, the full data transfer value is always applied by the end of each coupling step. This ramping behavior is demonstrated in Figure 46: Schematic of the Linear to Minimum Iterations ramping concept (p. 302) for the case where the minimum number of iterations specified is **5**.

When ramping using the **Linear to Minimum Iterations** or `Linear` option, if the minimum number of iterations is the same as the maximum number of iterations, then it is unlikely that the data transfer will converge. It is a best practice to set the maximum iterations to be larger than the minimum iterations.

Figure 46: Schematic of the Linear to Minimum Iterations ramping concept



Note:

An alpha-level setting allows you to revert to System Coupling's previous method of setting initial target-side reference values to program-specified constants. To do so, perform the following steps:

1. Enable alpha-level settings, as described in Activate System Coupling Hidden Features in the *System Coupling Beta Features* documentation.
2. Set `AnalysisControl.TargetInitializationOption` to ***`UseConstantValue`***.

For details about the constant values that will be applied, see "Initial Values Used for the Reference Target-Side Value" in the 2020 R1 *System Coupling User's Guide*.

Under-Relaxation Algorithm

Under-relaxation is used to improve convergence of the overall analysis. It works by limiting a potentially large variation of the target-side data between two successive coupling iterations. For each data transfer location (node), the following formula is applied:

$$\phi_{\text{Relaxed}} = \phi_{\text{Reference}} + \omega \left(\phi_{\text{Raw}} - \phi_{\text{Reference}} \right) \tag{5}$$

where

ϕ_{Relaxed} is the relaxed, target-side value.

$\phi_{\text{Reference}}$ is the reference target-side value. For coupling iterations within a coupling step, the reference target-side value is the final value from the last coupling iteration.

For the first iteration of the first step in a coupling run (either an initial or a restarted run), the reference target-side value is interpolated by System Coupling. For the first coupling iteration of all subsequent steps, the reference target-side value is the final value from the last coupling step.

ϕ_{Raw} is the raw, target-side value obtained from interpolation or from ramping (if applied).

Note that if you have applied both ramping and under-relaxation, the data are first ramped and then under-relaxed. In this case, $\phi_{\text{Raw}}=\phi_{\text{Ramped}}$ for the under-relaxation's raw target-side value.

ω is the under-relaxation factor (URF). In a transient analysis, in the first coupling iteration of every coupling step, the URF is overridden and set to 1, and so data transferred at this coupling iteration is not under-relaxed.

For data model details, see `RelaxationFactor` in the *System Coupling Settings and Commands Reference* manual.

Quasi-Newton Stabilization Algorithm

System Coupling offers the ***Quasi-Newton solution stabilization and acceleration method***, which is based on the *Interface Quasi-Newton – Inverse Least Squares (IQN-ILS)* [1] (p. 304) algorithm. The Quasi-Newton method can be used to address convergence issues in two-way System Coupling simulations, which can otherwise occur in the following contexts:

Thermal Analysis:

- For transfers of Temperature and Heat Rate quantity types, convergence instabilities may develop if the participant sending heat rate has a thermal resistance of the same order as, or smaller than, the participant sending temperature. Usually, a fluid region sends heat rate to a solid region, so this may occur if the fluid region has a small thickness or large conductivity relative to the solid region.
- Transferring Heat Transfer Coefficient and Convection Reference Temperature quantity types (if supported by the participants) may be more robust than transferring Heat Rate and Temperature quantity types, but with two drawbacks:
  - the transfer is non-conservative, and
  - convergence can be very slow, particularly with fine meshes (for example, if the CFD mesh has a small y+).

FSI Simulations:

- For transfers of Force and Incremental Displacement quantity types, reaching convergence can be challenging if the physics are strongly coupled — that is, a small change in the fluid force leads to a significant change in deformation or vice versa. This is particularly common when the solid body is slender or if the solid density is comparable to or smaller than the fluid density, leading to significant added mass effect.
- Participant solution stabilization (p. 402) may improve stability in this situation but may also require problem-dependent tuning.

The Quasi-Newton stabilization method improves convergence in these situations by applying an approximate Newton iteration to the coupling interface data. The inverse Jacobian-vector products associated with the residual of the interface data is approximated using a least squares model, as explained in [1] (p. 304).

Thermal-Electric Simulations:

For transfers of Loss (Heat Rate Density) and Temperature, convergence can be difficult in some scenarios such as a core loss which has a negative dependence on temperature.

For more information, see:

Using Quasi-Newton Stabilization

Available Stabilization Settings

Quasi-Newton Recommendations

Quasi-Newton Limitations

Bibliography

[1] Joris Degroote, Robby Haelterman, Sebastiaan Annerel, Peter Bruggeman, and Jan Vierendeels. Performance of partitioned procedures in fluid-structure interaction. *Computers and Structures* 88: 446-457, 2010. DOI=http://dx.doi.org/10.1016/j.compstruc.2009.12.006

Using Quasi-Newton Stabilization

Quasi-Newton stabilization is activated by setting `AnalysisControl.GlobalStabilization.Option` to ***`Quasi-Newton`***. With this setting, the software will choose which data transfers to stabilize automatically. In most cases, the default selections are appropriate.

If you are using System Coupling's GUI or CLI and it is necessary to override the defaults for individual data transfers, you can set `Stabilization.Option` to ***`None`*** or ***`Quasi-Newton`***, rather than the default ***`ProgramControlled`*** option.

For more information, see Available Stabilization Settings (p. 306).

For instructions on using Quasi-Newton stabilization in different coupling contexts, see:

Changing Stabilization Settings in the GUI

Changing Stabilization Settings in the CLI

Enabling Stabilization for System Coupling in Workbench

Changing Stabilization Settings in the GUI

To change Quasi-Newton stabilization settings in the **System Coupling** GUI, perform the steps described in the following sections:

Activating/Configuring Global Stabilization in the GUI

Activating Local Stabilization in the GUI

Activating/Configuring Global Stabilization in the GUI

To activate and configure global stabilization in the GUI, perform the following steps:

1. In the **Outline** tree's **Setup** branch, expand **Analysis Control**.
2. Click **Global Stabilization**.

   The global **Option** setting is shown below in the **Properties** pane. By default, it is set to **None**, indicating that global stabilization is inactive.
3. For the **Option** setting, select **Quasi-Newton**.

   Global stabilization is activated and additional settings are shown below.
4. Optionally, you can enter new values for the **Initial Relaxation Factor** and/or **Maximum Retained Time Step** settings.

When applicable, the **Messages** tab shows validation details, informational messages, and recommendations for optimizing stabilization settings.

Activating Local Stabilization in the GUI

To activate local stabilization in the GUI, perform the following steps:

1. Activate global stabilization, as described in Activating/Configuring Global Stabilization in the GUI (p. 304).
2. In the **Outline** tree's **Setup** branch, expand **Coupling Interface | Interface-<#> | Data Transfer | <*DataTransferName*>**.
3. Click **Stabilization**.

   The local **Option** setting is shown below in the **Properties** pane. By default, it is set to **Program Controlled**, indicating that System Coupling determines whether stabilization is applied to the parent data transfer.
4. For the **Option** setting, select one of the other available values:

   - **None**
   - **Quasi-Newton**

When stabilization is activated for a data transfer, it is controlled by the global **Initial Relaxation Factor** and **Maximum Retained Time Step** settings.

When applicable, the **Messages** tab shows validation details, informational messages, and recommendations for optimizing stabilization settings.

Changing Stabilization Settings in the CLI

To change Quasi-Newton stabilization settings in System Coupling's CLI, perform the steps described in the following sections.

Activating/Configuring Global Stabilization in the CLI

Activating Local Stabilization in the CLI

Activating/Configuring Global Stabilization in the CLI

To activate and configure global stabilization in the CLI, perform the following steps:

1. To activate global stabilization, set the global `Option` setting to ***`Quasi-Newton`***.

```
>>> DatamodelRoot().AnalysisControl.GlobalStabilization.Option = 'Quasi-Newton'
```

2. Optionally, you can change the values for the `InitialRelaxationFactor` and/or `MaximumRetainedTimeSteps` settings.

```
>>> sc.GlobalStabilization.MaximumRetainedTimeSteps = 2
>>> sc.GlobalStabilization.InitialRelaxationFactor = 0.05
```

Activating Local Stabilization in the CLI

To activate local stabilization in the CLI, perform the following steps:

1. Activate global stabilization, as described in Activating/Configuring Global Stabilization in the CLI (p. 306).

2. By default, the local `Option` setting is set to ***`ProgramControlled`***, indicating that System Coupling determines whether stabilization is applied to the parent data transfer.

   To activate local stabilization, set `Option` to ***`Quasi-Newton`***.

```
>>> DatamodelRoot().CouplingInterface[Interface-<#>']
.DataTransfer['<TargetVariableDisplayName>-<#>'].Stabilization.Option = 'Quasi-Newton'
```

   When stabilization is activated for a data transfer, it is controlled by the global `InitialRelaxationFactor` and `MaximumRetainedTimeStep` settings.

Enabling Stabilization for System Coupling in Workbench

To enable Quasi-Newton stabilization in Workbench, enable the `GlobalStabilization` expert setting.

System Coupling in Workbench supports only default global stabilization settings.

For more detailed information, see Expert Settings (p. 256).

Available Stabilization Settings

The following data model settings control the application of solution stabilization:

Global Stabilization Settings

Global stabilization is controlled by `AnalysisControl.GlobalStabilization.Option`, which can take the following values:

- ***None*** (default value)

  Global stabilization is not active.

- ***Quasi-Newton***

  Global stabilization is active. Whether a data transfer is included in the global stabilization is determined by the transfer's local stabilization option (p. 307), as described below.

  When global stabilization is active, the following additional controls are also available:

  – ***InitialRelaxationFactor***: Specifies the under-relaxation factor to be applied during the startup iteration (that is, before the Quasi-Newton algorithm starts modifying the data transfers) to ensure divergence does not occur during startup.

    Accepts real values ***0.0 x 1.0***. Default value is ***0.01***.

    The default relaxation factor is quite heavy. If the risk of immediate divergence is low, then you can often achieve faster convergence by increasing this value.

  – ***MaximumRetainedTimeSteps***: Available for transient analyses. Specifies the maximum number of previous time steps to be retained in the Quasi-Newton history.

    Accepts integer values greater than or equal to than ***1***. Default value is ***1***.

    The optimal value is problem-dependent. Some cases may achieve faster convergence by increasing the value (for example, to ***5***).

Local Stabilization Option

Available only for coupled analyses run in System Coupling's GUI or CLI.

Local stabilization is controlled per-transfer by the `CouplingInterface.DataTransfer.Stabilization.Option` setting, which becomes available when global stabilization is activated. This setting can take the following values:

- ***ProgramControlled*** (default value)

  The software chooses whether to include this data transfer in the global stabilization algorithm based on the physics and data transfer type.

- ***None***

  Stabilization is not applied to the data transfer.

- ***Quasi-Newton***

Stabilization is applied to this data transfer.

Note:

It is possible for multiple data transfers to be selected for stabilization, if there are multiple coupling interfaces, or if there are multiple data transfers on a single interface. Note that it is inappropriate to stabilize multiple transfers if those transfers are in opposite directions. For example, it is inappropriate to stabilize both Force and Displacement in an FSI calculation, or both Temperature and Heat Flow in a thermal calculation.

Quasi-Newton Recommendations

When applying the Quasi-Newton method, consider the following recommendations to ensure optimal stabilization:

- In most cases, you should activate stabilization by setting `GlobalStabilization` to ***`Quasi-Newton`*** and leaving the `Stabilization | Option` settings for the individual transfers set to ***`ProgramControlled`***.

  Full details are shown in System Coupling's transcript or by issuing the `PrintSetup()` command in the CLI or the GUI's **Command Console**.

- If you wish to force stabilization to be active for a data transfer not selected by the ***`Program-Controlled`*** option, then you may do so and leave other data transfers set to ***`Program-Controlled`***. The ***`ProgramControlled`*** algorithm will avoid invalid combinations.

- Care is required if you manually select ***`Quasi-Newton`*** for multiple individual transfers, as it is easy to select invalid combinations. (For example, it is inappropriate to stabilize both Temperature and Heat Rate in a thermal analysis, or Force and Incremental Displacement in an FSI analysis.)

- For a thermal analysis, Ansys recommends that you transfer Temperature and Heat Rate quantity types (not Heat Transfer Coefficient and Convection Reference Temperature), since these are conservative and usually converge faster when combined with stabilization.

- Strongly coupled FSI simulations may not converge deeply enough if the number of coupling iterations is too small. The default maximum number of coupling iterations per time step in a transient analysis is five. A larger value of ***`10`*** or ***`20`*** is recommended for such situations.

- Quasi-Newton stabilization should generally not be combined with ramping, relaxation, or participant stabilization.

- Quasi-Newton stabilization may not be effective if a participant solver does not converge deeply enough. This situation arises most frequently for CFD participants when the CFD solver is restricted to a small number of CFD iterations per coupling iteration. The number of required CFD iterations depends on the case and CFD solver settings (for example, with Fluent, the coupled solver typically converges better than segregated algorithms), but in general, at least 20 CFD iterations per coupling iteration are recommended.

Quasi-Newton Limitations

Currently, the Quasi-Newton algorithm has the following limitation:

- Quasi-Newton data from previous steps are not restarted when the value of the `Maximum-RetainedIterations` setting is greater than ***1***. This may change the convergence history, compared to a solution which is not restarted, but will not affect the converged result.
- Use of Quasi-Newton with point cloud regions is a beta feature.

System Coupling Input and Output Files

This section describes System Coupling's input and output files – the files it consumes and the files it produces during the setup and execution of a coupled analysis. For more information, see:

Files Used by System Coupling

Files Generated by System Coupling

Files Used by System Coupling

This section describes the files used by System Coupling during its execution:

Participant Setup File

Input File

Shutdown File

Participant Setup File

When applicable, coupling participants write a ***System Coupling Participant*** setup file (`*.scp`). Typically, these files are named according to the convention noted below, but other extensions can be used, as well.

`<ParticipantObject>.scp`

These XML-formatted files provide System Coupling with the participant-specific information needed to connect the participants to a coupling run.

Important:

The sole purpose of an `.scp` file is to allow the participant to be added to a coupled analysis (p. 121). When participants are added, the information from their `.scp` files populates the data model and then is saved, as part of the analysis state, to the Settings file (p. 351). From this point, the analysis draws its setup information from the Settings file, not from the participant `.scp` files.

Ansys recommends that you do not attempt to change participant settings by editing `.scp` files manually, as this may invalidate the data model and/or cause a mismatch between the analysis and its Settings file.

Also, note that reloading an `.scp` file to an existing analysis overwrites the setup details in the corresponding Settings file, so this should not be done unless you intend to start over with the analysis setup. If you do attempt to change participant settings by editing `.scp` files, ensure that edits are made before you start setting up the analysis (that is, add participants, create interfaces, and so on).

For an example, see Sample System Coupling Participant File (p. 320).

Note:

- The use of both `.sci` and `.scp` files in the same analysis is not supported.
- In the sections that follow, noted commands and queries can be run either in the System Coupling CLI in the **System Coupling** GUI's **Command Console**.

The `.scp` file includes the following main elements:

```
<CouplingParticipant>
 <ExecutionControl/> (p. 312)
 <CosimulationControl/> (p. 313)
</CouplingParticipant>
```

<ExecutionControl>

Container element for settings that control the execution of the participant. Provides the information needed to start the coupling participant.

Optional. If this element does not exist, System Coupling populates the `ExecutionControl` part of the data model with values based on the participant type.

Contains the following sub-elements:

```
<ExecutionControl>
 <InitialInput/> (p. 312)
 <Command/> (p. 313)
</ExecutionControl>
```

<InitialInput>

Defines the name of the solver input file to be used for the initial run of the coupled analysis.

Required if the parent `<ExecutionControl> (p. 312)` element exists.

Acceptable `<InitialInput>` values vary according to coupling participant:

- For an AEDT participant, it must refer to a solver input file (`.py`), which in turn refers to an Ansys Electronics Desktop project file (`.aedt`).
- For a CFX participant, it must refer to a CFX-Solver input (`.def` or `.res`) file.
- For a Mechanical participant, it must refer to a Mechanical APDL solver input file (`.dat`).
- For a Fluent participant, there are two options:
  - It can refer to a Fluent case file (`.cas`, `.cas.gz`, or `.cas.h5`), either with or without the extension. In this case, System Coupling generates its own Fluent journal script to read this case file.

– It can refer to a custom Fluent journal file (`.jou`). The journal file must have a `.jou` extension.

<Command>

Specifies the command that can be used to start the coupling participant automatically.

Optional. Note that this element is not included when a participant generates its System Coupling Participant (`.scp`) setup file because System Coupling can locate the participant's execution command based on other information from the file.

To obtain the command used to start a participant, run `GetExecutionCommand()` for the participant. For participant-specific details, see the participant's documentation on executing from the command line.

<CosimulationControl>

Container element for settings that control how the participant can act in a co-simulation. Provides the information necessary to set up couplings and to execute the participant in the co-simulation.

Required.

Contains the following sub-elements:

```
<CosimulationControl>
 <Type/> (p. 313)
 <Dimension/> (p. 313)
 <AnalysisType/> (p. 314)
 <DisplayName/> (p. 314)
 <RestartsSupported/> (p. 314)
 <Variables/> (p. 314)
 <Regions/> (p. 318)
</CosimulationControl>
```

<Type>

Specifies the type of participant.

Optional. When not specified, the ***DEFAULT*** type is used.

Possible values:

- ***DEFAULT*** (default value)
- ***AEDT***
- ***CFX***
- ***FLUENT***
- ***MAPDL***

<Dimension>

Specifies the dimension of the participant.

Optional.

Possible values:

- ***3D*** (default value)
- ***2D***

<AnalysisType>

Specifies the type of analysis the participant is running.

Optional. If not specified, defaults to ***Steady***.

Possible values:

- ***Steady*** (default value)
- ***Transient***

<DisplayName>

Specifies the participant name to be displayed in user-facing communications.

Optional. When not specified, the name of the participant object in the data model is used.

<RestartsSupported>

Indicates whether the participant supports restarts for this type of coupled analysis.

If any participant in a coupled analysis does not support restarts, then restarts are disabled for the analysis as whole. A participant's restart capabilities may vary according to the type of analysis being performed.

Optional. If not specified, either in the participant's coupling `.scp` file or in the data model, then the default value is determined by the participant's `<ParticipantType>` value, as described below:

Possible values:

- ***False***: Indicates that the participant does not support restarts.

  If not specified, then this is the default value for participants of type ***FORTE*** or ***DEFAULT***.

- ***True***: Indicates that the participant supports restarts.

  If not specified, then this is the default value for participants *not* of type ***FORTE*** or ***DEFAULT***.

<Variables>

Container element for the set of variables that are available for coupled simulations.

Optional. If no output/input variables are defined, then the participant is ignored.

Contains one or more <Variable> (p. 315) sub-elements:

```
<Variables>
 <Variable> (p. 315)
</Variables>
```

<Variable>

Defines a variable that is available for coupled simulations. Nested under <Variables> (p. 314).

Optional. If no output/input variables are defined, then the participant is ignored.

Contains the following sub-elements:

```
<Variable>
 <Name> (p. 315)
 <DisplayName> (p. 315)
 <QuantityType> (p. 315)
 <TensorType> (p. 316)
 <IsExtenstive> (p. 316)
 <Location> (p. 316)
</Variable>
```

<Name>

Specifies the variable name to be used internally by System Coupling and the participant.

Required.

<DisplayName>

Specifies the variable name to be displayed in user-facing communications.

Optional. If not specified, the variable's data model <Name> (p. 315) is used.

<QuantityType>

Indicates the quantity type of the data provided by the variable.

Required when values are not specified for the variable's <TensorType> (p. 316) and/or <IsExtensive> (p. 316) elements.

Otherwise, optional. If a quantity type is not available (that is, either the <QuantityType> element is left blank or set to ***Unspecified***), then:

- Values for the <TensorType> and/or <IsExtensive> elements are required.
- Any variables with the same <TensorType> can be coupled.

Possible values:

- ***Unspecified***

- ***Convection Reference Temperature***
- ***Force***
- ***Incremental Displacement***
- ***Heat Rate***
- ***Heat Transfer Coefficient***
- ***Temperature***

<TensorType>

Indicates the tensor type of the variable being transferred.

Required when the <QuantityType> (p. 315) is not available (that is, either the <QuantityType> element is left blank or set to ***Unspecified***).

Otherwise, optional. If a value is not specified for the <TensorType> element, then System Coupling infers this value, either from the variable's <Name> (for MAPDL participants) or from its <QuantityType> (for all other participant types).

Possible values:

- ***Scalar***
- ***Vector***

<IsExtensive>

Indicates whether the variable is extensive. A property is extensive if its value depends on the size of the system.

Not available for FMU participants.

Required when the <QuantityType> (p. 315) is not available (that is, either the <QuantityType> element is left blank or set to ***Unspecified***).

Otherwise, optional. If a value is not specified for the <IsExtensive> element, then System Coupling infers this value, either from the variable's <Name> (for MAPDL participants) or from its <QuantityType> (for all other participant types).

Possible values:

- ***True***: The variable is extensive.
- ***False***: The variable is intensive.

<Location>

Defines the mesh location on which data exists.

Optional. If not specified, defaults to ***Node***.

Possible values:

- ***Element***
- ***Node*** (default value)

<Parameters>

Container element for the set of parameters that are available for coupled simulations.

Optional. When no parameters are defined, parameters are not added to the data model.

Contains one or more <Parameter> sub-elements:

```
<Parameters>
 <Parameter> (p. 317)
</Parameters>
```

<Parameter>

Defines a parameter that is available for coupled simulations. Nested under <Parameters>.

Required when the <Parameters> element exists.

Contains the following sub-elements:

```
<Parameter>

 <Name> (p. 317)
 <DisplayName> (p. 317)
 <TensorType> (p. 317)
</Parameter>
```

<Name>

Specifies the region name to be used by System Coupling and the participant.

Required.

<DisplayName>

Specifies the region name to be displayed in user-facing communications.

Optional. If not specified, the parameter object's data model <Name> (p. 317) is used.

<TensorType>

Indicates the tensor type of the parameter being transferred. The only value allowed is *Scalar*.

<InputParameters>

Container element for the set of parameters for which the parameter can receive data.

Optional. If no input variables are defined, then no data transfers may be created with the parameter as a target. Contains one or more <Parameter> sub-elements.

```
<InputParameters>
 <Parameter> (p. 318)
</InputParameters>
```

<Parameter>

Defines an input parameter for which the parameter can receive data. Must correspond to the <Name> of a parameter defined under the <Parameters> element.

Optional. If no input parameters are defined, then no data transfers may be created with the parameter as a target.

<OutputParameters>

Container element for the set of parameters for which the parameter can send data.

Optional. If no input variables are defined, then no data transfers may be created with the parameter as a target. Contains one or more <Parameter> sub-elements.

```
<OutputParameters>
 <Parameter> (p. 318)
</OutputParameters>
```

<Parameter>

Defines an output parameter for which the parameter can receive data. Must correspond to the <Name> of a parameter defined under the <Parameters> element.

Optional. If no output parameters are defined, then no data transfers may be created with the parameter as a source.

<Regions>

Container element for the set of regions that are potentially available for coupled simulations.

Optional. When no regions are defined, the participant is ignored.

Contains one or more <Region> sub-elements:

```
<Regions>
 <Region> (p. 319)
</Regions>
```

<Region>

Defines a region that is potentially available for coupled simulations. Nested under <Regions> (p. 318).

Optional. When no regions are defined, the participant is ignored.

Contains the following sub-elements:

```
<Regions>
 <Topology> (p. 319)
 <Name> (p. 319)
 <DisplayName> (p. 319)
 <OutputVariables> (p. 319)
 <InputVariables> (p. 320)
</Regions>
```

<Topology>

Defines the spatial dimensionality of the data being transferred to/from the region.

Required.

Possible values:

- ***Surface***
- ***Volume***

<Name>

Specifies the region name to be used by System Coupling and the participant.

Required.

<DisplayName>

Specifies the region name to be displayed in user-facing communications.

Optional. If not specified, the region object's data model <Name> (p. 319) is used.

<OutputVariables>

Container element for the set of output variables for which the region can send data.

Optional. If no output variables are defined, then no data transfers may be created with the region as a source.

Contains one or more <Variable> (p. 320) sub-elements.

```
<OutputVariables>
 <Variable> (p. 320)
</OutputVariables>
```

<Variable>

Defines an output variable for which the region can send data. Must correspond to the <Name> of a variable defined under the <Variables> (p. 313) element.

Optional. If no output variables are defined, then no data transfers may be created with the region as a source.

<InputVariables>

Container element for the set of input variables for which the region can receive data.

Optional. If no input variables are defined, then no data transfers may be created with the region as a target.

Contains one or more <Variable> (p. 320) sub-elements.

```
<InputVariables>
 <Variable> (p. 320)
</InputVariables>
```

<Variable>

Defines an input variable for which the region can receive data. Must correspond to the <Name> of a variable defined under the <Variables> (p. 314) element.

Optional. If no input variables are defined, then no data transfers may be created with the region as a target.

Sample System Coupling Participant File

The following is an example .scp file for the Electronics Desktop participant in a coupled analysis.

```
<CouplingPartipant>
   <ExecutionControl>
      <InitialInput>FreeBurningArc_Maxwell3DDesign1_SystemCouplingSetup2.py</InitialInput>
   </ExecutionControl>
   <CosimulationControl>
      <Type>AEDT</Type>
      <AnalysisType>Transient</AnalysisType>
      <RestartsSupported>False</RestartsSupported>
      <DisplayName>Ansys Electronics Desktop</DisplayName>
      <Variables>
         <Variable>
            <Name>Loss</Name>
            <QuantityType>Heat Rate</QuantityType>
            <TensorType>Scalar</TensorType>
            <IsExtensive>True</IsExtensive>
            <Location>Element</Location>
         </Variable>
         <Variable>
            <Name>Temperature</Name>
            <QuantityType>Temperature</QuantityType>
            <TensorType>Scalar</TensorType>
            <IsExtensive>False</IsExtensive>
            <Location>Element</Location>
         </Variable>
         <Variable>
```

```
            <Name>Conductivity</Name>
            <QuantityType>Electrical Conductivity</QuantityType>
            <TensorType>Scalar</TensorType>
            <IsExtensive>False</IsExtensive>
            <Location>Element</Location>
         </Variable>
         <Variable>
            <Name>LorentzForce</Name>
            <QuantityType>Force</QuantityType>
            <TensorType>Vector</TensorType>
            <IsExtensive>True</IsExtensive>
            <Location>Element</Location>
         </Variable>
         <Variable>
            <Name>SurfaceLoss</Name>
            <QuantityType>Heat Rate</QuantityType>
            <TensorType>Scalar</TensorType>
            <IsExtensive>True</IsExtensive>
            <Location>Element</Location>
         </Variable>
      </Variables>
      <Parameters>
         <Parameter>
            <Name>Current</Name>
            <DisplayName>Current</DisplayName>
         </Parameter>
         <Parameter>
            <Name>Voltage</Name>
            <DisplayName>Voltage</DisplayName>
         </Parameter>
      </Parameters>
      <InputParameters>
         <Parameter>Current</Parameter>
      </InputParameters>
      <OutputParameters>
         <Parameter>Voltage</Parameter>
      </OutputParameters>
      <Regions>
         <Region>
            <Topology>Volume</Topology>
            <Name>Region15</Name>
            <DisplayName>Cathode</DisplayName>
            <OutputVariables>
               <Variable>Loss</Variable>
               <Variable>LorentzForce</Variable>
               <Variable>SurfaceLoss</Variable>
            </OutputVariables>
            <InputVariables>
               <Variable>Temperature</Variable>
            </InputVariables>
         </Region>
         <Region>
            <Topology>Volume</Topology>
            <Name>Region24</Name>
            <DisplayName>Anode</DisplayName>
            <OutputVariables>
               <Variable>Loss</Variable>
               <Variable>LorentzForce</Variable>
               <Variable>SurfaceLoss</Variable>
            </OutputVariables>
            <InputVariables>
               <Variable>Temperature</Variable>
            </InputVariables>
         </Region>
         <Region>
            <Topology>Volume</Topology>
            <Name>Region105</Name>
            <DisplayName>Arc_region_inner</DisplayName>
            <OutputVariables>
               <Variable>Loss</Variable>
               <Variable>LorentzForce</Variable>
```

```
               <Variable>SurfaceLoss</Variable>
            </OutputVariables>
         </Region>
         <Region>
            <Topology>Volume</Topology>
            <Name>Region317</Name>
            <DisplayName>Arc_region_Separate2</DisplayName>
            <OutputVariables>
               <Variable>Loss</Variable>
               <Variable>LorentzForce</Variable>
               <Variable>SurfaceLoss</Variable>
            </OutputVariables>
         </Region>
         <Region>
            <Topology>Volume</Topology>
            <Name>Region343</Name>
            <DisplayName>Arc_start_region</DisplayName>
            <OutputVariables>
               <Variable>Loss</Variable>
               <Variable>LorentzForce</Variable>
               <Variable>SurfaceLoss</Variable>
            </OutputVariables>
            <InputVariables>
               <Variable>Conductivity</Variable>
               <Variable>Temperature</Variable>
            </InputVariables>
         </Region>
         <Region>
            <Topology>Volume</Topology>
            <Name>Region763</Name>
            <DisplayName>Arc_region_inner1</DisplayName>
            <OutputVariables>
               <Variable>Loss</Variable>
               <Variable>LorentzForce</Variable>
               <Variable>SurfaceLoss</Variable>
            </OutputVariables>
            <InputVariables>
               <Variable>Conductivity</Variable>
               <Variable>Temperature</Variable>
            </InputVariables>
         </Region>
         <Region>
            <Topology>Volume</Topology>
            <Name>Region826</Name>
            <DisplayName>Arc_region_Separate3</DisplayName>
            <OutputVariables>
               <Variable>Loss</Variable>
               <Variable>LorentzForce</Variable>
               <Variable>SurfaceLoss</Variable>
            </OutputVariables>
            <InputVariables>
               <Variable>Conductivity</Variable>
               <Variable>Temperature</Variable>
            </InputVariables>
         </Region>
      </Regions>
   </CosimulationControl>
</CouplingPartipant>
```

Input File

When a coupled analysis setup is completed in Workbench the **System Coupling** system generates a ***System Coupling Input*** file (`scInput.sci`)file.

Note:

Values in the `.sci` file are set by System Coupling and/or participants. These are typically not edited manually — and if edited at all, care should be taken to not invalidate the data model.

The `.sci` file includes the following elements:

<Transfers>

<ExecutionControl>

<Analysis>

<Participants>

<Transfers>

The `<Transfers>` element contains details used to define the data transfers between any static and co-simulation coupling participants.

In this element, you can set the ***Count*** (an integer representing the total number of data transfers) as an attribute.

For each coupling transfer, the following properties are available:

<Name> (string)

Name of the participant. This is the name with which the participant identifies itself to System Coupling. This corresponds to the **Component ID** which is unique to a specific system's **Solution** cell in the Workbench user interface.

<Type> (integer attribute)

Type of coupling participant.

Possible values:

- ***0***: Co-simulation
- ***1***: Static data

<DisplayName> (wide string)

Display name of the participant, as provided in the Workbench user interface.

<FilePath> (string)

Full path to the primary file used to access source data from a static data participant.

<SupportsCouplingIterations> (Boolean)

Controls whether the co-simulation participant supports the execution of multiple coupling iterations per coupling step.

<UnitSystem>

Unit system used.

<Regions> (options below are applicable to an individual region)

For each region, the following properties are available:

<Name> (string)

Name of the region (intrinsic to the participant).

<DisplayName> (wide string)

Display name of the region, as provided in the Workbench user interface.

<TopologicalDimensionality> (integer)

Geometry type of the region.

Possible values:

- ***2***: Surface
- ***3***: Volume

<Variables> (options below are applicable to an individual variable)

<Name> (string)

Name of the variable (intrinsic to the participant).

<DisplayName> (wide string)

Display name of the variable, as in the Workbench user interface.

<PhysicalType> (string)

Physical type of the variable. Possible values include:

- ***Length***
- ***Force***

<BaseUnits> (strings denoting base units for all data of noted physical type)

Possible values:

- ***Angle*** (string)
- ***ChemicalAmount*** (string)

- ***Current*** (string)
- ***Length*** (string)
- ***Luminance*** (string)
- ***Mass*** (string)
- ***SolidAngle*** (string)
- ***Temperature*** (string)
- ***Time*** (string)

<ExecutionControl>

The <ExecutionControl> element contains details used for the execution of the participant. Contains the following sub-elements:

<IntermediateResultsFileOutput>

The <IntermediateResultsFileOutput> element contains properties that specify the frequency at which intermediate result files, which can be used for restarts, are written by System Coupling.

<FrequencyOption> (integer)

Frequency with which intermediate restart files are generated.

Possible values:

- ***0***: Every coupling step
- ***1***: Coupling step interval

<StepInterval> (integer)

Available when <FrequencyOption> is set to ***StepInterval***. Coupling step interval at which intermediate result files are generated. For example, using a step interval of 3, results are generated at steps 3, 6, 9, ...

<MappingSettings>

May be reported in the .sci file but is not used by System Coupling.

<Analysis>

The <Analysis> element contains details used to define the coupled analysis. In this element, you can set the following:

<AnalysisType> (integer)

Defines the nature of the sequential steps used in coupling co-simulation participants.

Possible values:

- ***0***: General
- ***1***: Transient

<Initialization>

Defines the initial time for the coupled analysis.

<Option> (integer)

Possible values:

- ***0***: Program Controlled

  System Coupling makes the most appropriate choice of an initial time value. (default value)

- *1*: Start Time

  System Coupling overrides the initial/start time for the analysis with the value specified as part of <Time>.

<Time> (double)

Available when Initialization <Option> is set to *1*. Initial time for the coupled analysis.

<Duration>

Defines the duration of the coupled analysis.

<Option> (integer)

Possible values:

- ***0***: Number of Steps
- ***1***: End Time

<NumberOfSteps> (integer)

Available if no end-time requirements exist for co-simulation participants.

<Time> (double)

Final time of coupling analysis.

<Step>

MaximumIterations (integer)

Maximum number of coupling iterations allowed per coupling step.

<MinimumIterations> (integer)

Minimum number of coupling iterations allowed per coupling step.

<Size> (double)

Size of the coupling step when it is associated with a time (this is done for transient analyses, as size is measured in seconds).

<Option> (integer)

Possible values:

- ***1***: Coupling step size. Used for transient analyses.
- ***0***: Non-dimensional step size. Used for general analyses.

<Unit System>

Unit system used.

<Participants>

The read-only <Participants> element contains information obtained through connections to upstream solver systems.

In this element, you can view the <Count> (an integer representing the number of connected participants).

Note:

Depending upon the type of participant (co-simulation or static data), some of the options may not be applicable.

For each connected participant, you can view the following:

Type (integer attribute)

Type of coupling participant.

Possible values:

- ***0***: Co-simulation
- ***1***: Static data

<Name> (string)

Name of the participant. This is the name with which the participant identifies itself to System Coupling. This corresponds to the **Component ID**, which is unique to a specific system's **Solution** cell in the Workbench user interface.

<DisplayName> (wide string)

Display name of the participant, as provided in the Workbench user interface.

<ParticipantType> (string)

Indicates the application participating in the coupled analysis.

Possible values:

- ***CFX***
- ***FLUENT***
- ***MAPDL***
- ***DEFAULT***: Used for static participants, such when a static data file is used instead of a solver. Used when no participant type is specified. (default value)

<FilePath> (string)

Full path to the primary file used to access source data from a static data participant.

<SupportsCouplingIterations> (Boolean)

Controls whether the co-simulation participant supports the execution of multiple coupling iterations per coupling step.

<UnitSystem>

Unit system used.

<Regions> (options below are applicable to an individual region)

For each region, the following properties are available:

<Name> (string)

Name of the region (intrinsic to the participant).

<DisplayName> (wide string)

Display name of the region, as provided in the Workbench user interface.

<TopologicalDimensionality> (integer)

Geometry type of the region.

Possible values:

- ***0***: Undefined
- ***1***: Point
- ***2***: Curve

- ***3***: Surface
- ***4***: Volume

`Variables` (options below are applicable to an individual variable)

For each variable, the following properties are available:

`<Name>` (string)

Name of the variable (intrinsic to the participant).

`<DisplayName>` (wide string)

Display name of the variable, as provided in the Workbench user interface.

`<PhysicalType>` (string)

Physical type of the variable.

Possible values include:

- ***`Length`***
- ***`Force`***

`<BaseUnits>` (strings denoting base units for all data of noted physical type)

Possible values include:

- ***`Angle`*** (string)
- ***`ChemicalAmount`*** (string)
- ***`Current`*** (string)
- ***`Length`*** (string)
- ***`Luminance`*** (string)
- ***`Mass`*** (string)
- ***`SolidAngle`*** (string)
- ***`Temperature`*** (string)
- ***`Time`*** (string)

Shutdown File

The Shutdown file is a text file named `scStop` which indicates whether the run should be interrupted or aborted. When System Coupling finds this file in its working directory, it shuts down as soon as possible according to the contents of the file. The Shutdown file can be used to stop coupling runs that are executed using either System Coupling's CLI or System Coupling in Workbench.

System Coupling's CLI

A coupled analysis that is run from the command line using the `Solve()` command performs the entire solution straight through (as opposed to an analysis that is run using interactive solve commands (p. 159), which provide the ability to pause, interact with, and shut down the run at designated points). However, you can either interrupt or abort such an analysis by creating a Shutdown file in System Coupling's working directory.

System Coupling in Workbench

When a coupled analysis is stopped by clicking either **Abort** or **Interrupt**, a Shutdown file is automatically generated in System Coupling's working directory. However, you can also create a Shutdown file in the working directory manually.

Create the Shutdown file according to how the analysis is to be stopped. In general, the file contains one or two lines:

- The first line contains a single integer, which indicates whether to interrupt (***0***) or abort (***1***) the run.
- The second line is optional and contains text that indicates the reason for terminating the run. This information is recorded in System Coupling's Transcript.

Tip:

You can abort a run by simply creating an empty `scStop` file. This is most useful when you want to abort the run as abruptly as possible.

The Shutdown file does not have a file extension. When creating the file manually, ensure that the file does not have an extension so that System Coupling can find it.

When the analysis is interrupted, System Coupling completes the current coupling step or iteration and signals the coupling participants that the execution has ended. This causes System Coupling and participants to shut down cleanly and create restart points for the end of the run. An example Shutdown file to interrupt the run is shown below.

```
0
Interrupt this run because resources are needed for another analysis.
```

When the analysis is aborted, System Coupling signals the coupling participants to stop the run as quickly as possible. No restart points are created for the end of the run. Note, however, that any previously created restart points are retained.

Once the analysis has been stopped, the Shutdown file is automatically removed from System Coupling's working directory.

Files Generated by System Coupling

This section describes the files generated by System Coupling during its execution:

GUI Server File

When you start System Coupling in GUI Server mode — whether started from the GUI or from the command line with the `--guiserver` argument — it writes a ***System Coupling GUI Server*** file (`sycGui.srv`) is written to its working directory.

System Coupling writes the server port and host information to the GUI Server file in the following format, where "***12345***" is the server port and "***0.0.0.0***" is the host information. This information is used to connect/reconnect an instance of the **System Coupling** GUI to the running solution.

Figure 47: Example GUI Server file

```
guiserver=12345@0.0.0.0
```

An instance of System Coupling started in GUI Server mode can be connected to only one GUI at a time. The GUI Server file remains in the working directory while System Coupling is running there, preventing additional instances of the **System Coupling** GUI from connecting with the active coupling session.

When System Coupling exits, whether naturally (for example, due to solve or script completion or because of an *Abort* or *Interrupt* operation) or unnaturally (for example, as the result of a handled exception), the GUI Server file is removed from the working directory.

Important:

The GUI Server file should not be removed manually unless System Coupling was terminated unexpectedly and cleanup is required. For details on handling this scenario, see Known Issues and Limitations (p. 47).

Solution Lock File

When a coupled analysis solution is started by any method, System Coupling writes a ***System Coupling Solution Lock*** file (`sycSolve.lock`) to its working directory.

This file remains in the working directory and disables the *Save*, *Solve*, and any of the *Open* or *Step* operations while the solution is in progress, preventing other instances of System Coupling from interfering its execution.

When the solution stops, whether naturally (for example, a completed solution, the end of a script, or an *Interrupt* or *Abort* operation) or via a handled exception, the Solution Lock file is removed from the working directory.

Important:

The Solution Lock file should not be removed manually unless System Coupling was terminated unexpectedly and cleanup is required. For details on handling this scenario, see Known Issues and Limitations (p. 47).

Transcript and Log File

System Coupling writes coupled analysis information to two output formats:

System Coupling Transcript

During the execution of the run, coupled analysis information is written dynamically to the Transcript, so you can use it both to monitor the analysis during the solution process and to review coupling results after the solution is finished. When a coupled analysis is restarted, the content for the restarted run is appended to the end of the existing file.

- For System Coupling in Workbench, Transcript output is written to the **System Coupling** tab's **Solution Information** view.
- For System Coupling run in one of its user interfaces, Transcript output is written to the CLI or the GUI's **Command Console** and **SC Transcript** tabs.

System Coupling Log File

During the execution of the run, coupled analysis information from the Transcript is written to the Log file (`scLog.scl`) file, which is created in the **SyC** output folder in System Coupling's working directory.

The Transcript and Log file include the following sections:

Summary of Coupling Setup

Analysis Initialization

Coupled Solution

Shut Down

Tip:

The ability to set the level of precision used in the Transcript/Log file is available as beta functionality. For more information, see Set Transcript/Log Precision in the *System Coupling Beta Features* documentation.

Note:

In the sections that follow, output examples are selected to illustrate variances in output, so are not all generated from the same coupled analysis.

Summary of Coupling Setup

Provides information about the coupled analysis setup, preceded by the time at which the Transcript output was started.

When you run the `Solve()` command, this section is written to the command console at the beginning of the coupling run. To write it to the console at other times during the run, you can run the `PrintSetup` command.

Note that when the **Summary of Coupling Setup** is printed using the `PrintSetup()` command, all variables and regions defined for the coupling participants are listed in this section. When the **Summary of Coupling Setup** is printed at the beginning of a solve, however, only the regions and variables actively involved in a data transfer are included.

Figure 48: Summary of Coupling Setup section header

```
================================================================================
Transcript output started: 2021-07-29 12:06:22


================================================================================
+==============================================================================+
|                                                                              |
|                          Summary of Coupling Setup                           |
|                                                                              |
+==============================================================================+
================================================================================
```

Summary of Coupling Setup contents:

Library

Provides information about any library objects (that is, expressions, expression functions, or transformation reference frames that are used in the coupled analysis. If no library objects are defined, then this section is not generated.

Figure 49: Library section

```
+=========================================================================+
|                                Library                                  |
+=========================================================================+
|                                                                         |
| Expression:If Condition                                                 |
|    Expression Name :                                       IfCondition  |
|    Expression String : 2 if count >= 8 else 2.25 if count == 7 else 2.5 if  |
| Expression:End Time                                                     |
|    Expression Name :                                           EndTime  |
|    Expression String :                                           1 [s]  |
| Expression:Timestep Size                                                |
|    Expression Name :                                       TimestepSize |
|    Expression String :                                           0.1 [s] |
| Expression:count                                                        |
|    Expression Name :                                             count  |
|    Expression String :                                            Step  |
| Reference Frame: Frame-1                                                |
|    Option :                                          ByTransformation   |
|    Parent Reference Frame :                        GlobalReferenceFrame |
|    Transformation Order :                                  Rotation-1   |
|    Transformation:Rotation-1                                            |
|       Angle :                                                -90 [deg]  |
|       Axis :                                                     YAxis  |
|       Option :                                                 Rotation |
| Instancing: Instance-1                                                  |
|    Instances For Mapping :                                           2  |
|    Instances In Full Circle :                                        8  |
|    Reference Frame :                               GlobalReferenceFrame |
+=========================================================================+
```

Coupling Participants

Provides information about each participant solver.

Participants are ordered alphabetically by display name. Display names are also used for regions and variables.

For details about corresponding settings in the data model, see `CouplingParticipant` in the *System Coupling Settings and Commands Reference* manual.

Figure 50: Coupling Participants section

```
+=========================================================================+
|                        Coupling Participants (2)                        |
+=========================================================================+
|    Fluid Flow (CFX)                                                     |
+-------------------------------------------------------------------------+
|       Internal Name :                                      Solution 3   |
|       Participant Type :                                          CFX   |
|       Participant Display Name :                     Fluid Flow (CFX)   |
|       Dimension :                                                 3D    |
|       Input Parameters :                                          []    |
|       Output Parameters :                                         []    |
|       Participant Analysis Type :                               Steady  |
|       Restarts Supported :                                        True  |
|       Variables (2)                                                     |
|          Variable : Force                                               |
|             Internal Name :                                Total Force |
|             Quantity Type :                                      Force  |
|             Location :                                           Node   |
```

```
|             Participant Display Name :                                Force |
|             Data Type :                                                Real |
|             Tensor Type :                                            Vector |
|             Is Extensive :                                             True |
|                                                                             |
|          Variable : Mesh_Displacement                                       |
|             Internal Name :                                Mesh Displacement |
|             Quantity Type :                           Incremental Displacement |
|             Location :                                                 Node |
|             Participant Display Name :                     Mesh Displacement |
|             Data Type :                                                Real |
|             Tensor Type :                                            Vector |
|             Is Extensive :                                            False |
|       Regions (2)                                                           |
|          Region : coupling                                                  |
|             Internal Name :                                         coupling |
|             Topology :                                              Surface |
|             Input Variables :                              [Mesh Displacement] |
|             Output Variables :                                   [Total Force] |
|                                                                             |
|          Region : wall                                                      |
|             Internal Name :                                             wall |
|             Topology :                                             Surface |
|             Input Variables :                                           [] |
|             Output Variables :                                          [] |
|       Update Control                                                        |
|          Option :                                          ProgramControlled |
|       Execution Control                                                     |
|          Option :                                                UserDefined |
|          Working Directory :                                           SOLN1 |
|          Executable :                                   SOLN1\participant2 |
|          Additional Arguments :                                         None |
|          Parallel Fraction :                                        1.00e+00 |
|          Initial Input :                                            case.def |
+-----------------------------------------------------------------------------+
|    Static Structural                                                        |
+-----------------------------------------------------------------------------+
|       Internal Name :                                              Solution |
|       Participant Type :                                              MAPDL |
|       Participant Display Name :                        Static Structural |
|       Dimension :                                                      3D |
|       Input Parameters :                                               [] |
|       Output Parameters :                                              [] |
|       Participant Analysis Type :                                  Steady |
|       Restarts Supported :                                           True |
|       Variables (3)                                                         |
|          Variable : Force                                                   |
|             Internal Name :                                            FORC |
|             Quantity Type :                                           Force |
|             Location :                                                 Node |
|             Participant Display Name :                                Force |
|             Data Type :                                                Real |
|             Tensor Type :                                            Vector |
|             Is Extensive :                                             True |
|                                                                             |
|          Variable : Force_Density                                           |
|             Internal Name :                                            FDNS |
|             Quantity Type :                                           Force |
|             Location :                                              Element |
|             Participant Display Name :                                Force |
|             Data Type :                                                Real |
|             Tensor Type :                                            Vector |
|             Is Extensive :                                            False |
|                                                                             |
|          Variable : Incremental_Displacement                                |
|             Internal Name :                                            INCD |
|             Quantity Type :                           Incremental Displacement |
|             Location :                                                 Node |
|             Participant Display Name :                Incremental Displacement |
|             Data Type :                                                Real |
|             Tensor Type :                                            Vector |
```

```
|             Is Extensive :                                          False |
|       Regions (1)                                                         |
|          Region : Fluid Solid Interface                                   |
|             Internal Name :                                        FSIN_1 |
|             Topology :                                             Surface |
|             Input Variables :                                  [FORC, FDNS] |
|             Output Variables :                                       [INCD] |
|       Update Control                                                      |
|          Option :                                        ProgramControlled |
|       Execution Control                                                   |
|          Option :                                              UserDefined |
|          Working Directory :                                          SOLN |
|          Executable :                                  SOLN\participant1 |
|          Additional Arguments :                                       None |
|          Parallel Fraction :                                    1.00e+00 |
|          Initial Input :                                              None |
|          Additional Restart Input File :                              None |
+===========================================================================+
```

Analysis Control

Provides information on settings that define and control the behavior of the coupled analysis.

For details about corresponding settings in the data model, see `AnalysisControl` in the *System Coupling Settings and Commands Reference* manual.

Figure 51: Analysis Control section

```
+===========================================================================+
|                            Analysis Control                               |
+===========================================================================+
|    Analysis Type :                                                 Steady |
|    Optimize If One Way :                                             True |
|    Allow Simultaneous Update :                                      False |
|    Partitioning Algorithm :                       SharedAllocateMachines |
|    Global Stabilization                                                   |
|       Option :                                                       None |
+===========================================================================+
```

Coupling Interfaces

Provides information about each coupling interface.

Interfaces are ordered alphabetically by display name. Display names are also used for participants, data transfers, regions, and variables.

For details about corresponding settings in the data model, see `CouplingInterface` in the *System Coupling Settings and Commands Reference* manual.

Figure 52: Coupling Interfaces section

```
+===========================================================================+
|                         Coupling Interfaces (1)                           |
+===========================================================================+
|    interface-1                                                            |
+---------------------------------------------------------------------------+
|       Internal Name :                                          interface-1 |
|       Side                                                                |
|          Side: One                                                        |
```

```
|             Coupling Participant :                              Solution 3 |
|             Region List :                                       [coupling] |
|             Reference Frame :                       GlobalReferenceFrame |
|             Instancing :                                            None |
|          Side: Two                                                       |
|             Coupling Participant :                                Solution |
|             Region List :                                         [FSIN_1] |
|             Reference Frame :                       GlobalReferenceFrame |
|             Instancing :                                            None |
|       Data Transfers (2)                                                 |
|          DataTransfer : Data Transfer                                    |
|             Internal Name :                                  Data Transfer |
|             Suppress :                                               False |
|             Target Side :                                              Two |
|             Option :                                        UsingVariable |
|             Source Variable :                                 Total Force |
|             Target Variable :                                        FORC |
|             Ramping Option :                                         None |
|             Relaxation Factor :                                  1.00e+00 |
|             Convergence Target :                                 1.00e-02 |
|             Mapping Type :                                   Conservative |
|          DataTransfer : Data Transfer 2                                  |
|             Internal Name :                                Data Transfer 2 |
|             Suppress :                                               False |
|             Target Side :                                              One |
|             Option :                                        UsingVariable |
|             Source Variable :                                        INCD |
|             Target Variable :                             Mesh Displacement |
|             Ramping Option :                                         None |
|             Relaxation Factor :                                  1.00e+00 |
|             Convergence Target :                                 1.00e-02 |
|             Mapping Type :                                ProfilePreserving |
|             Unmapped Value Option :                           Nearest Value |
|       Mapping Control                                                    |
|          Stop If Poor Intersection :                                 False |
|          Face Alignment :                                 ProgramControlled |
|          Absolute Gap Tolerance :                                0.0 [m] |
|          Relative Gap Tolerance :                               1.00e+00 |
+==========================================================================+
```

Solution Control

Provides information on settings that define and control the behavior of the solution. For details about corresponding settings in the data model, see `SolutionControl` in the *System Coupling Settings and Commands Reference* manual.

Figure 53: Solution Control section

```
+==========================================================================+
|                            Solution Control                              |
+==========================================================================+
|    Duration Option :                                      NumberOfSteps |
|    Number Of Steps :                                                  2 |
|    Minimum Iterations :                                               5 |
|    Maximum Iterations :                                               5 |
|    Use IP Address When Possible :                                  True |
+==========================================================================+
```

Output Control

Provides information on settings that define and control the behavior of the coupled analysis.

For details about corresponding settings in the data model, see `OutputControl` in the *System Coupling Settings and Commands Reference* manual.

Figure 54: Output Control section

```
+=============================================================================+
|                             Output Control                                  |
+=============================================================================+
|    Option :                                                        LastStep |
|    Generate CSV Chart Output :                                        False |
|    Write Initial Snapshot :                                            True |
|    Results                                                                  |
|       Option :                                          ProgramControlled   |
|       Include Instances :                               ProgramControlled   |
|       Type                                                                  |
|          Option :                                             EnsightGold   |
+=============================================================================+
```

One-Way Workflow Message

If the `OptimizeIfOneWay` setting is enabled and the data transfers are one-way, a message indicating this is written to the Transcript.

Figure 55: One-Way Workflow Messages section

```
+=============================================================================+
|                       One-way Workflow Optimizations                        |
+=============================================================================+
|                                                                             |
| The data transfers in this simulation define a one-way simulation. The      |
| participant execution order will be determined automatically, with a       |
| single coupling iteration for each step.                                    |
|                                                                             |
+=============================================================================+
```

Pre-Solution Warnings and Messages

Prior to the solution details, System Coupling prints any validation warnings about issues that could affect execution. When applicable, it prints messages indicating that hidden features are activated and noting which hidden settings have been changed from their default values at any time during the course of the analysis.

Figure 56: Pre-Solution Warnings and Messages section

```
+-----------------------------------------------------------------------------+
| Warnings were found during data model validation.                           |
+-----------------------------------------------------------------------------+
| Restart output is only requested at last step. To enable restarts from      |
|     intermediate points, use another 'OutputControl' option.                |
+-----------------------------------------------------------------------------+

+-----------------------------------------------------------------------------+
| Hidden features are available.                                              |
+-----------------------------------------------------------------------------+
| Beta: 'DisableInitialSnapshotWrite' in AnalysisControl                      |
```

```
+------------------------------------------------------------------------+
```

EnSight Output Display Name Modifications

When variable or region display names are changed in the EnSight-formatted output files, a table shows the original and modified display name for each item.

Figure 57: EnSight Output Display Name Modifications section

```
+========================================================================+
|                  EnSight Output Display Name Modifications             |
+========================================================================+
|                           Modified Variable Names                      |
+------------------------------------------------------------------------+
| Region: Fluid Flow (Fluent) :        |                                 |
|   deforming_wall                     |                                 |
|   Original:                          | Modified:                       |
|   force__EV                          | force_1__EV                     |
|   force__EV                          | force_2__EV                     |
|   force__EV                          | force_3__EV                     |
|   force__EV                          | force_4__EV                     |
|   force__EV                          | force_5__EV                     |
+========================================================================+
```

Partitioning Information

Provides parallel execution information. It lists all available machines and cores, as well as which of those machines and cores are available to each of the participants. This information is available when one of the following conditions are met:

- System Coupling is running with `--cnf` command line arguments explicitly specified.
- System Coupling is running *outside* of the HPC job scheduler and `-t` command line arguments explicitly specified.
- System Coupling is running *inside* of the HPC job scheduler environment.

Figure 58: Partitioning Information section

```
+========================================================================+
|                         Partitioning Information                       |
+========================================================================+
| Machine Name                         | Core Count                      |
+------------------------------------------------------------------------+
|                                                                        |
| All Available Machines                                                 |
+------------------------------------------------------------------------+
|   AAPjLoa2Vxm7cJj                    | 4                               |
+------------------------------------------------------------------------+
|                                                                        |
| Machines Available to Participants                                     |
+------------------------------------------------------------------------+
| Fluid Flow (Fluent)                                                    |
|   AAPjLoa2Vxm7cJj                    | 4                               |
+------------------------------------------------------------------------+
| Default                                                                |
|   AAPjLoa2Vxm7cJj                    | 4                               |
```

```
+=========================================================================+
```

Execution Information

Provides information related to the start-up and execution of the coupled analysis, such as the machines being used and details about intranode communications.

Note:

This information is written only to the Transcript and does not appear in the Log file.

Figure 59: Execution Information section

```
+=========================================================================+
|                           Execution Information                         |
+=========================================================================+
|                                                                         |
| System Coupling                                                         |
|   Command Line Arguments:                                               |
|     --guiserver -m cosimgui                                             |
|   Working Directory:                                                    |
|     D:\CoilAndCoreMechanical                                            |
|                                                                         |
| Ansys Electronics Desktop                                               |
|   Execution Command:                                                    |
|     "C:\Program Files\AnsysEM\AnsysEM25.1\Win64\ansysedt.exe" -ng -features |
|     =SF6694_NON_GRAPHICAL_COMMAND_EXECUTION -scport 60038 -schost MYACHINE. |
|     win.ansys.com -scname "AEDT-1" -runscript "CoilAndCore_Maxwell3DDesign_ |
|     SystemCouplingSetup1.py"                                            |
|   Working Directory:                                                    |
|     D:\CoilAndCoreMechanical\Maxwell                                    |
|                                                                         |
| MAPDL Transient Thermal                                                 |
|   Execution Command:                                                    |
|     "C:\Program Files\ANSYS Inc\v251\ansys\bin\winx64\ANSYS251.exe" -o MAPD |
|     L-2.out -b nolist -s noread -scport 60038 -schost MYMACHINE.win.ansys.c |
|     om -scname "MAPDL-2" -scid="15192_CHQKFLETCHE.win.ansys.com" -sclic=600 |
|     39@127.0.0.1 -i "CoilAndCore.dat"                                   |
|   Working Directory:                                                    |
|     D:\\CoilAndCoreMechanical\Mechanical                                |
|                                                                         |
+=========================================================================+
```

Build Information

Provides build information for System Coupling and each participant, with participants ordered alphabetically by display name.

Figure 60: Build Information section

```
+=========================================================================+
|                            Build Information                            |
+-------------------------------------------------------------------------+
| System Coupling                                                         |
|   2025 R1: Build ID: 5e0ec62 Build Date: 1 May 2022 09:04:46            |
| Ansys Electronics Desktop                                               |
```

```
|   2025.1.0                                                                  |
| MAPDL Transient Thermal                                                     |
|   Mechanical APDL Release 2025 R1          Build 25.1BETA UP20200823        |
|   DISTRIBUTED WINDOWS x64  Version                                          |
+=============================================================================+
```

Analysis Initialization

Provides meshing statistics, interpolation diagnostics (when applicable), and mapping diagnostics from the initialization of the coupled analysis.

Figure 61: Analysis Initialization section header

```
===============================================================================
+=============================================================================+
|                                                                             |
|                           Analysis Initialization                           |
|                                                                             |
+=============================================================================+
===============================================================================
```

Analysis Initialization contents:

Mesh Statistics

Each time participant meshes are updated, System Coupling writes a **Mesh Statistics** section to the Transcript. This section provides meshing details for each meshed region and for the coupled analysis as a whole, as follows:

For each participant region:

- Region name
- Region cell statistics (the counts of cell zones, cells, and cells of each type)

> **Note:**
>
> When performing volume-to-volume or surface-to-volume mapping, System Coupling detects the number of cells with warped faces and records the number of them here.
>
> Warped-face handling also affects volume-to-surface mapping, but the effects in this case are very small.

- Region face statistics: (the counts of face zones, faces, and faces of each type)
- Region area or volume
- Region extent (minimum and maximum)

For the coupled analysis:

Total mesh element statistics (the counts of cells, faces, and nodes across all regions)

If instancing and/or reference frame transformations are defined for the analysis, mesh statistics are reported for the original (non-instanced/untransformed) geometries.

Figure 62: Mesh Statistics section

```
+-------------------------------------------------------------------------+
|                            MESH STATISTICS                              |
+-------------------------------------------------------------------------+
| Participant: DEFAULT-1                                                  |
|   Number of cell regions                                            1 |
|     Number of cells                                            61 757 |
|       Tetrahedral                                              61 757 |
|     Number of faces                                           128 552 |
|     Volume (m3)                                             4.193e-06 |
|   Bounding Box (m)                                                      |
|     Minimum                          [ 0.000e+00  0.000e+00 -5.000e-03] |
|     Maximum                          [ 4.000e-02  2.000e-02  5.000e-03] |
|                                                                         |
| Participant: DEFAULT-2                                                  |
|   Number of cell regions                                            1 |
|     Number of cells                                            16 755 |
|       Hexahedral                                               16 650 |
|       Wedge                                                       105 |
|       With warped faces                                         5 055 |
|     Number of faces                                            52 417 |
|     Volume (m3)                                             4.193e-06 |
|   Bounding Box (m)                                                      |
|     Minimum                          [ 0.000e+00  0.000e+00 -5.000e-03] |
|     Maximum                          [ 4.000e-02  2.000e-02  5.000e-03] |
|                                                                         |
| Total                                                                   |
|   Number of cells                                              78 512 |
|   Number of faces                                             180 969 |
|   Number of nodes                                              32 132 |
+-------------------------------------------------------------------------+
| Instancing and/or Reference Frame transformations are defined for this  |
| analysis. Note that the mesh statistics report non-instanced/untransformed |
| geometries.                                                             |
+-------------------------------------------------------------------------+
```

Mapping Summary

Each time participant meshes are mapped/remapped (as described in Mapping Process (p. 285)), System Coupling writes a **MAPPING SUMMARY** to the Transcript. Depending on analysis details, this summary may be written at coupled analysis initialization and/or at the start of the first coupling iteration of each coupling step for which participant meshes have changed.

The **MAPPING SUMMARY** provides mapping diagnostics for all mapped coupling interfaces, with interfaces and data transfers ordered alphabetically by display name. For each data transfer on the mapped coupling interfaces, the following diagnostics are provided:

- **Mapped Volume [%]**: Indicates the percentage of the total volume that was successfully mapped for the source mesh and for the target mesh.
- **Mapped Area [%]**: Indicates the percentage of the total area that was successfully mapped for the source mesh and for the target mesh.
- **Mapped Elements [%]**: Indicates the percentage of source elements that map to target elements, and vice versa.
- **Mapped Nodes [%]**: Indicates the percentage of source nodes that were successfully mapped for the source mesh and for the target mesh.

Figure 63: Mapping Summary section

```
+-------------------------------------------------------------------------+
|                            MAPPING SUMMARY                              |
+-------------------------------------------------------------------------+
|                                  |      Source            Target       |
+-------------------------------------------------------------------------+

| Interface-1                      |                                     |
|   Heat Rate Density              |                                     |
|     Mapped Volume [%]            |        100               100        |
|     Mapped Elements [%]          |        100               100        |
|     Mapped Nodes [%]             |        100               100        |
|   Temperature                    |                                     |
|     Mapped Volume [%]            |        25                100        |
|     Mapped Elements [%]          |        14                100        |
|     Mapped Nodes [%]             |         6                98         |
| Interface-2                      |                                     |
|   Heat Rate Density              |                                     |
|     Mapped Volume [%]            |        100               100        |
|     Mapped Elements [%]          |        100               100        |
|     Mapped Nodes [%]             |        100               100        |
|   Temperature                    |                                     |
|     Mapped Volume [%]            |        23                100        |
|     Mapped Elements [%]          |        12                100        |
|     Mapped Nodes [%]             |         6                99         |
+-------------------------------------------------------------------------+
```

Note:

- **If between 99% and 100%** of either the source-side or target-side locations were mapped, then this is shown in the diagnostics as **>99%** and all other values are rounded to the nearest whole number.
- When motion is defined on coupled bodies, Ansys recommends that you monitor the System Coupling Transcript's **Mapping Summary** sections during execution to ensure that the bodies move consistently throughout the analysis.

Reference Frame and Instancing Information

When a case includes a coupling interface with reference frames and/or instancing defined on at least one side, the **Reference Frame and Instancing Information** section is written to the Transcript.

This section provides information on interface reference frames and instances, non-zero transformation values, and bounding boxes. If an interface has both rotation and translation, the rotation is applied first.

Figure 64: Reference Frame and Instancing section

```
+------------------------------------------------------------------------+
|                REFERENCE FRAME AND INSTANCING INFORMATION               |
+------------------------------------------------------------------------+
| Interface-1                                                            |
|    Side One                                                            |
|       Number of Instances :                                          2 |
|       Transformation                                                   |
|          Translation (m) :       [ 1.000000000,  0.000000000,  0.500000000] |
|          Rotation Angle (rad) :                                1.570796327 |
|          Axis :                  [-1.000000000,  0.000000000,  0.000000000] |
|       Bounding Box (m)                                                 |
|          Minimum :                     [ 9.000e-01, -1.000e-01,  5.000e-01] |
|          Maximum :                     [ 2.00                            |
|    Side Two                                                            |
|       Number of Instances :                                          1 |
|       Bounding Box (m)                                                 |
|          Minimum :                     [ 9.000e-01, -1.000e-01,  5.000e-01] |
|          Maximum :                     [ 2.000e+00,  1.000e+00,  5.000e-01] |
+------------------------------------------------------------------------+
| Information: If a rotation and translation appear in the same reference     |
| frame, the rotation is applied first.                                  |
+------------------------------------------------------------------------+
```

Transfer Diagnostics

When interpolation is performed during coupled analysis initialization, initial data transfer interpolation diagnostics are written to the **Transfer Diagnostics** section of the Transcript's **Analysis Initialization** block.

A section for each target participant indicates the data transfer it receives and shows the initial interpolation diagnostics for that data transfer, as follows:

- For extensive variables (which are mapped using a conservative algorithm), the **Sum** reports the net sum of values leaving the source and entering the target. Diagnostics are reported per-component for vector and complex variables.
- For intensive variables (which are mapped using a profile-preserving algorithm), the **Weighted Average** reports a weighted average of values on the source and on the target. Diagnostics are reported per-component for vector and complex variables. The averages are weighted by area for surface regions and by volume for volume regions.

Figure 65: Transfer Diagnostics section

```
+------------------------------------------------------------------------+
|                          Transfer Diagnostics                          |
+------------------------------------------------------------------------+
| Fluid Flow (Fluent)               |                                    |
|   Interface: Interface-1          |                                    |
|     displacement                  |                                    |
|       Weighted Average x          |     0.00E+00          0.00E+00     |
|       Weighted Average y          |     0.00E+00          0.00E+00     |
```

```
|       Weighted Average z            |     0.00E+00          0.00E+00        |
+-------------------------------------------------------------------------------+
| MAPDL Transient                     |                                       |
|   Interface: Interface-1            |                                       |
|     Force                           |                                       |
|       Sum x                         |    -2.73E-11          0.00E+00        |
|       Sum y                         |     1.25E+04          1.25E+04        |
|       Sum z                         |     1.09E+02          1.09E+02        |
+-------------------------------------------------------------------------------+
```

Coupled Solution

Provides information about the coupled solution and its execution.

Figure 66: Coupled Solution section header

```
================================================================================
+==============================================================================+
|                                                                              |
|                             Coupled Solution                                 |
|                                                                              |
+==============================================================================+
================================================================================
```

Coupled Solution contents:

Coupling Step

Coupling Step

Provides information for each coupling step, including the simulation time and details for each coupling step completed. Within the coupling step, the following information is provided, with participants ordered according to update sequence:

Mapping Summary

Coupling Iteration

Note that the coupling step's **SIMULATION TIME** is provided for transient analyses.

Figure 67: Coupling Step section header

```
+==============================================================================+
| COUPLING STEP = 1                        SIMULATION TIME = 1.00000E+00 [s] |
+------------------------------------------------------------------------------+
```

Coupling Step contents:

Mapping Summary

Coupling Iteration

Mapping Summary

Each time participant meshes are mapped/remapped (as described in Mapping Process (p. 285)), System Coupling writes a **MAPPING SUMMARY** to the Transcript.

If mapping occurs at the beginning of a coupling step, then the summary is written to the Transcript at the beginning of the step's first iteration.

Figure 68: Mapping Summary at the Coupling Step level

```
-------------------------------------------------------------------------------+
|                          COUPLING ITERATION = 1                               |
+-------------------------------------------------------------------------------+
|                              MAPPING SUMMARY                                  |
+-------------------------------------------------------------------------------+
|                                   |      Source            Target           |
+-------------------------------------------------------------------------------+
| Interface-1                       |                                         |
|   Heat Rate Density               |                                         |
|     Mapped Volume [%]             |        100               100            |
|     Mapped Elements [%]           |        100               100            |
|     Mapped Nodes [%]              |        100               100            |
|   Temperature                     |                                         |
|     Mapped Volume [%]             |        25                100            |
|     Mapped Elements [%]           |        14                100            |
|     Mapped Nodes [%]              |         6                98             |
| Interface-2                       |                                         |
|   Heat Rate Density               |                                         |
|     Mapped Volume [%]             |        100               100            |
|     Mapped Elements [%]           |        100               100            |
|     Mapped Nodes [%]              |        100               100            |
|   Temperature                     |                                         |
|     Mapped Volume [%]             |        23                100            |
|     Mapped Elements [%]           |        12                100            |
|     Mapped Nodes [%]              |         6                99             |
+-------------------------------------------------------------------------------+
```

Coupling Iteration

Provides details for each coupling iteration in the step, with participants listed in order of the update sequence. For each iteration, the following information is provided:

Participant data transfers:

A section for each participant indicates the data transfer it receives and whether the data transfer converged during the iteration.

- **Data transfer diagnostics:**
  - For each data transfer, the **RMS Change** in transfer values reports progress toward specified convergence targets.

> **Note:**
>
> The **RMS Change** diagnostic is not reported for transfers involving participant variables with complex number values.

  - For extensive variables (which are mapped using a conservative algorithm), the **Sum** reports the net sum of values leaving the source and entering the target. Diagnostics are reported per-component for vector and complex variables.

  - For intensive variables (which are mapped using a profile-preserving algorithm), the **Weighted Average** reports a weighted average of values on the source and on the target. Diagnostics are reported per-component for vector and complex variables. The averages are weighted by area for surface regions, and by volume for volume regions.

- **Participant ordering:**

  - Participant data transfers, whether for individual participants or participants grouped for simultaneous solution, are ordered chronologically by participant update sequence, as shown in Figure 69: Coupling Iteration section (p. 347).

  - Data transfers for grouped participants that are solving simultaneously are not separated by a line in the output, as shown in Figure 70: Coupling Iteration section with simultaneous-solve participants (p. 348).

Participant solution diagnostics:

Indicates whether each participant's solution converged (it solved its equations) during the iteration.

Possible values:

- **Converged**: Used for iterative solutions. Indicates that the solution has converged during the iteration. (Not applicable to AEDT participants.)
- **Complete**: Used for direct solutions and all AEDT solutions (that is, both iterative and direct). Indicates that the solution has been completed for the iteration.
- **Convergence Not Evaluated**: Indicates that the participant's output was not monitored during the iteration.
- **Not Updated**: Indicates that the solution was not updated during the iteration, as specified by the participant's `UpdateControl` settings, as described in the *System Coupling Settings and Commands Reference* manual.

Display names are used for interfaces, participants, and data transfers.

Figure 69: Coupling Iteration section

```
+-------------------------------------------------------------------------+
|                          COUPLING ITERATION = 2                         |
+-------------------------------------------------------------------------+
| MAPDL 1                          |                                      |
|   Interface: Interface-1         |                                      |
|     Heat Rate Density            |         Not yet converged            |
|       RMS Change                 |     1.25E+00          1.10E+00       |
|       Sum                        |    -2.03E-01         -1.40E+01       |
+-------------------------------------------------------------------------+
| MAPDL 2                          |                                      |
|   Interface: Interface-2         |                                      |
|     Heat Rate Density            |         Not yet converged            |
|       RMS Change                 |     1.30E+00          1.28E+00       |
```

```
|       Sum                          |    -3.22E-01         -1.98E+01        |
+------------------------------------------------------------------------------+
| Fluent                             |                                       |
|   Interface: Interface-1           |                                       |
|     Temperature                    |         Not yet converged             |
|       RMS Change                   |     3.80E-04          3.80E-04        |
|       Weighted Average             |     1.02E+02          1.02E+02        |
|   Interface: Interface-2           |                                       |
|     Temperature                    |             Converged                 |
|       RMS Change                   |     8.64E-05          8.64E-05        |
|       Weighted Average             |     1.11E+02          1.11E+02        |
+------------------------------------------------------------------------------+
| Participant solution status        |                                       |
|   MAPDL 1                          |             Converged                 |
|   MAPDL 2                          |             Converged                 |
|   FLUENT                           |             Converged                 |
+==============================================================================+
```

Figure 70: Coupling Iteration section with simultaneous-solve participants

```
+------------------------------------------------------------------------------+
|                          COUPLING ITERATION = 2                              |
+------------------------------------------------------------------------------+
| MAPDL 1                            |                                       |
|   Interface: Interface-1           |                                       |
|     Heat Rate Density              |         Not yet converged             |
|       RMS Change                   |     1.25E+00          1.10E+00        |
|       Sum                          |    -2.03E-01         -1.40E+01        |
| MAPDL 2                            |                                       |
|   Interface: Interface-2           |                                       |
|     Heat Rate Density              |         Not yet converged             |
|       RMS Change                   |     1.30E+00          1.28E+00        |
|       Sum                          |    -3.22E-01         -1.98E+01        |
+------------------------------------------------------------------------------+
| Fluent                             |                                       |
|   Interface: Interface-1           |                                       |
|     Temperature                    |         Not yet converged             |
|       RMS Change                   |     3.80E-04          3.80E-04        |
|       Weighted Average             |     1.02E+02          1.02E+02        |
|   Interface: Interface-2           |                                       |
|     Temperature                    |             Converged                 |
|       RMS Change                   |     8.64E-05          8.64E-05        |
|       Weighted Average             |     1.11E+02          1.11E+02        |
+------------------------------------------------------------------------------+
| Participant solution status        |                                       |
|   MAPDL 1                          |             Converged                 |
|   MAPDL 2                          |             Converged                 |
|   FLUENT                           |             Converged                 |
+==============================================================================+
```

Shut Down

Provides information about the shutdown of System Coupling, including a list of the restart points created and timing details for the analysis.

Figure 71: Shut Down section header

```
================================================================================
```

```
+=========================================================================+
|                                                                         |
|                              Shut Down                                  |
|                                                                         |
+=========================================================================+
===========================================================================
```

Shut Down contents:

Available Restart Points

Provides a table of the restart points generated during the execution of the analysis, whether from the current run or previous runs. The corresponding Restart files are named according to the analysis type and the types of participants involved, as described in Restart Files (p. 351).

Figure 72: Available Restart Points section with step-based restart points

```
+=========================================================================+
|                         Available Restart Points                        |
+=========================================================================+
| Restart Point                       | File Name                         |
+-------------------------------------------------------------------------+
| Coupling Step 1                     | Restart_step1.h5                  |
| Coupling Step 2                     | Restart_step2.h5                  |
| Coupling Step 3                     | Restart_step3.h5                  |
| Coupling Step 4                     | Restart_step4.h5                  |
+=========================================================================+
```

Figure 73: Available Restart Points section with iteration-based restart points

```
+=========================================================================+
|                         Available Restart Points                        |
+=========================================================================+
| Restart Point                       | File Name                         |
+-------------------------------------------------------------------------+
| Coupling Iteration 1                | Restart_iter1.h5                  |
| Coupling Iteration 5                | Restart_iter5.h5                  |
| Coupling Iteration 10               | Restart_iter10.h5                 |
| Coupling Iteration 11               | Restart_iter11.h5                 |
+=========================================================================+
```

Timing Summary

In the **Timing Summary**, Individual participants are ordered alphabetically by display names and are listed before groups of simultaneous-solve participants.

The following timing details are provided:

Total Time

Records the total time of the coupled analysis, from when all participants connect to when they disconnect.

Coupling Participant Time

- For each ungrouped coupling participant, records its total processing time.
- For each participant group, lists its members and records the total processing time for the group (that is, the sum of processing times for all member participants).

Coupling Engine Time

Records the processing time used by the coupling engine for each phase of the coupled analysis, as follows:

- **Solution Control** is the time used by System Coupling's controller process to set up and advance the co-simulation to convergence.
- **Mesh Import** is the time used to import mesh regions from the participants and repartition into System Coupling's compute nodes.
- **Mapping Setup** is the time used to compute the mapping weights used to interpolate solution data.
- **Mapping** is the time used to interpolate solution data from sources to targets.
- **Numerics** is the time used for other numerical algorithms, such as Quasi-Newton acceleration, ramping, relaxation, and reconstructing solution data from nodes to elements.
- **Miscellaneous** is time used that is not covered by the other categories. For example, time spent by System Coupling to send data to and receive data from participants falls under this category.
- **Total** is the total processing time used by the coupling engine.

The summary is followed by messages indicating whether the run completed successfully and the time at which the Transcript output was stopped.

Figure 74: Timing Summary section

```
+=========================================================================+
|                           Timing Summary [s]                            |
+=========================================================================+
| Total Time :                                              4.92756E+02 |
| Coupling Participant Time                                             |
|    CFX :                                                  3.30890E+02 |
|    MAPDL Transient :                                      4.50147E+01 |
|    Total :                                                3.75905E+02 |
| Coupling Engine Time                                                  |
|    Solution Control :                                     8.67179E+01 |
|    Mesh Import :                                          5.67175E-02 |
|    Mapping Setup :                                        1.03958E-02 |
|    Mapping :                                              1.04814E-01 |
|    Numerics :                                             3.48588E-01 |
|    Misc. :                                                2.96124E+01 |
```

```
|    Total :                                                          1.16851E+02 |
+=============================================================================+

+=============================================================================+
|                 System coupling run completed successfully.                 |
+=============================================================================+

Transcript output stopped: 2021-07-26 16:31:08
===============================================================================
```

Settings File

When the data model state or simulation state is saved (using the *Save* or *Save Snapshot* operations or as part of a System Coupling setup exported from Workbench, analysis settings are written to a ***System Coupling Settings*** file. By default, the file is written to the **SyC** directory in the coupled analysis directory. When the state is saved using the `Save()` command, you can specify an alternate location by using the `FilePath` argument.

Important:

The Settings file reflects the state of the data model at the time the coupled was saved. To ensure consistency between the contents of the Settings file and the setup used by the coupled analysis, Ansys recommends that you do not attempt to change participant settings by editing the `.scp` files manually.

If data model changes are introduced and the *Save* operation is repeated then the new Settings file replaces the previous one. Also, if an `.scp` file is reloaded to an existing analysis, then the setup details in the corresponding Settings file are overwritten.

Restart Files

For a coupled analysis run in one of System Coupling's user interfaces, results are written to a ***System Coupling Restart*** file at the end of the last coupling step or iteration completed. By default, these files are written to the **SyC** subdirectory in System Coupling's working directory.

Restart files are named according to the convention `Restart_step<#>.h5` or `Restart_iter<#>.h5`, where "#" is the index of the corresponding coupling step or iteration.

Note:

Restart files named according to previous conventions (that is, `Results_step<#>|iter<#>.h5` or `Results_<#>.h5`) are still supported. System Coupling can read these files but names all new files according to the new conventions.

By default, when a restart point is created (p. 144) by any method, a Restart file containing the information necessary to restart the coupled analysis at the end of the last completed step or iteration is written to the **SyC** subdirectory in System Coupling's working directory.

When a coupled analysis stops, either because it completed or was shut down, a listing of the Restart files created is written to the Available Restart Points (p. 349) table at the end of the Transcript\Log file.

Restart files may correspond either to coupling iterations or coupling steps, depending on the analysis type and whether the participants involved support the creation of restart points for iterations, as follows:

Step-based restart files:

- Produced for:
  - Transient analyses
  - Steady-state analyses that include a participant that can only create restart points per step (that is, which have a `ParticipantType` of ***AEDT***, ***CFX***, ***MAPDL***, or ***EXTERNALDATA***)
- Named according to the convention `Restart_step<#>.h5`, where "#" is the index of the coupling step

Iteration-based restart files:

- Produced for:
  - Steady-state analyses that include only participants that can create restart points per iteration (that is, which **do not** have a `ParticipantType` of ***AEDT***, ***CFX***, ***MAPDL***, or ***EXTERNALDATA***)
- Named according to the convention `Restart_iter<#>.h5`, where "#" is the index of the coupling iteration

When you restart an analysis, by default the Restart file associated with the last solved coupling step or iteration is used. If you want to begin from an earlier restart point, then you can use the following methods to specify the restart point to be used:

- From the CLI, run the `Open()` command with the `CouplingStep` or `CouplingIteration` argument.
- From the GUI, use the **File > Open at step** or **File > Open at iteration** options.

Note, however, that when restarting from an earlier point, all the later restart points stored in the same directory are removed (that is, the Restart files are deleted). To keep restart data for one of these later coupling steps or iterations, save the analysis state to a snapshot before starting participants. For more information, see Working with Coupled Analysis Snapshots (p. 135).

For more detailed information on restart points and how to use them in one of System Coupling's user interfaces, see Restarting a Coupled Analysis (p. 142).

Export Setup Log File

When a System Coupling setup is exported as described in Exporting a System Coupling Setup (p. 273), the Export Setup Log file (`exportSetup.log`) file is written to System Coupling's working directory.

This file provides summary information for all the items related to the coupled analysis that were processed in the **Export** operation, such as Workbench systems, participant folders, and files. It includes the path of the coupled analysis working specified/created for the export and verifies whether the export completed successfully.

Ansys EnSight Results Files

System Coupling writes EnSight-compatible output to the **SyC/Results** folder in its working directory. Postprocessing files are always generated at initialization. By default, the subsequent postprocessing files are generated at the same frequency as restart points, as defined by the `OutputControl.Option` setting (which itself defaults to generating a single restart point at the end of the coupling run).

Note:

EnSight output will not be created if all interfaces in the analysis contain sides without regions defined. However, results are written for the interface sides containing regions.

To customize how and when EnSight postprocessing files are generated, you can use the settings defined under `OutputControl.Results`.

The EnSight-formatted postprocessing files generated by System Coupling are described below:

EnSight Command file:

A single `Results.enc` EnSight Command file is written at for the coupled analysis. It contains all the data for the analysis — that is, it references all the relevant output files, including the case files for all processors used to execute the solution.

When loading coupling results either from System Coupling's GUI or CLI or when starting EnSight from the command line, Ansys recommends that you use the `.enc` file generated for the coupled analysis.

For more information on playing the EnSight Command file upon startup, see Loading EnSight Gold Results with EnSight Startup (p. 212).

For more information on EnSight Command files, see Command Files in the *Ansys EnSight User Manual*.

Case files:

A Case file (`*.case`) in EnSight Gold format is written for each process that System Coupling spawns for the analysis. Each file contains only the data generated by the corresponding process.

When opening Gold-formatting coupling results in an existing instance of EnSight, Ansys recommends that you use the Case files generated for the coupled analysis. Ensure that you load all the Case files generated for the analysis so that the full set of results is available.

Note:

By default, System Coupling runs two processes on two compute nodes (that is, **`-s2`**). For more detailed information, see System Coupling Parallel Options in the *System Coupling Settings and Commands Reference* manual.

Case files are named according to the following conventions:

When a single System Coupling process is used:

- The file is named `Results.case`.
- This is the only instance where a single Case file contains all data for the analysis (and so could be used to load the full set of results into EnSight).

When multiple System Coupling processes are used:

- Files are named according to the convention `Results_<#>.case`, where "*#*" is the index of the associated process.
- Because System Coupling spawns two processes by default, it writes a results file for each one: `Results_0.case` and `Results_1.case`.

For more information on Case files, see EnSight Gold Case File Format in the *Ansys EnSight User Manual*.

Dynamic Visualization Store files:

When EnSight's live visualization capabilities are enabled, the Dynamic Visualization Store (DVS) writes a DVS-formatted file (`*.dvs`) which is used to export the co-simulation's results to EnSight.

Geometry data files:

Geometry data files have a `.geo` extension. Geometry results are written for each restart point and at the end of the run.

Named according to the convention `Results_<######>.geo`, where "*######*" is the six-digit coupling step index.

Variable data files:

Variable data files have a `.dat` extension. For each variable, coupling step results are written for each restart point and at the end of the run.

Named according to the convention `Results_<V>_<######>.dat`, where "*V* " is the variable transferred and "*######*" is the six-digit coupling step index.

Note:

To facilitate postprocessing, System Coupling provides a **total-displacement** variable which accumulates calculations for all incremental displacement variables defined for the **Coupling Interface** object. For analyses with one or more transfers of Incremental Displacement quantities, a single **total-displacement** variable will hold the accumulated totals of all incremental displacement variables.

Multi-step data files:

Filenames file:

EnSight's ***Filenames*** file is named `Results.filenames`. Individual file that tells EnSight how to complete the filenames.

This file has ***n*** lines, where "***n***" is the number of steps in the analysis. Each line in the file lists a step number.

Time file:

The `Results.time` file is an individual file that provides the time (in seconds) for transient runs.

This file is the same length as the `*.filenames` file, with ***n*** lines, where "***n***" is the number of steps in the analysis.

Units file:

The `Results.xml` file contains specifications for common units for physical variables.

Server File

The ***System Coupling Server*** file (`scServer.scs`) is written to System Coupling's working directory if any participant is not started automatically. This text file contains the information needed to manually connect participants to System Coupling.

The file contains the following lines of data:

- Coupling server's port and host, separated by an "@" character.

  Note:

  If the coupling server host reports an unexpected name, review the notes in Participant connection issues (p. 355) for information on how to resolve this.

- Element containing the number of coupling participants connected to System Coupling, including the automatically generated unique and display names for each. In the System Coupling environment:

- Unique names are used by System Coupling to identify and connect the participants.
- Display names are shown in user-facing communications such as files and console output.

Example 54: Example Server file

```
12345@CouplingServer.mydomain.com `             !Coupling server's port and host
2                                               !Number of participants
CFX-1                                           !Participant 1 Unique Name
CFX Solver Display Name                         !Participant 1 Display Name
Ansys                                           !Participant 2 Unique Name
Ansys Solver Display Name                       !Participant 2 Display Name
```

Note:

When the participants are started and instructed to connect to System Coupling, they must connect using unique names (for example, and `Ansys` in the example above).

Functional Mock-Up Unit (FMU) Co-Simulation Participants

System Coupling supports the use of co-simulation **Functional Mock-up Units (FMUs)** as coupling participants.

Note:

- Currently, model exchange FMUs are not supported.
- FMU-based coupling participants are referred to as "co-simulation slaves" in the FMI documentation.

When an FMU is integrated as a coupling participant, it:

- serves as a container for a third-party function or physics code, and
- communicates with System Coupling through the Functional Mock-up Interface (FMI) library routines.

Unlike other co-simulation participants, FMU participants are integrated into System Coupling using the Functional Mock-up Interface (FMI). Only the FMI 2.0 standard is supported. Participant capabilities are accessed by System Coupling coupled analyses via an FMU. An FMU is a file provided from an external tool that supports FMI for co-simulation.

Both types of participant interact with System Coupling in very similar ways and share many of the same coupling-specific properties in the System Coupling data model. However, FMU participants also have a number of capabilities and behaviors that are significantly different from those of Ansys participants. For more information, see Differences between FMU and Ansys Participants (p. 358).

Important:

This documentation assumes that you have some familiarity with co-simulation FMUs and can produce an FMU 2.0 file suitable for use in a System Coupling coupled analysis. For detailed information, see:

- FMI Functional Mock-up Interface: https://fmi-standard.org
- Functional Mock-up Interface for Model Exchange and Co-Simulation: https://svn.modelica.org/fmi/branches/public/specifications/v2.0/FMI_for_ModelExchange_and_CoSimulation_v2.0.pdf

For more information on FMU co-simulation participants, see:

Differences between FMU and Ansys Participants

FMU participants differ from Ansys participants in the following ways:

- FMU input/output variables and parameters are restricted to single scalar values and do not have quantity types.
- Expressions are not supported for data transfers from solvers to FMUs.
- Regarding data model settings:
  - For co-simulations involving an FMU participant, the `CouplingInterface` singleton does not include `MappingControl` settings.
  - FMU participants have `CouplingParticipant` settings that are significantly different from those of Ansys participants. For details, see FMU `CouplingParticipant` Data Model Settings (p. 362).

Supported System Coupling Functionality

System Coupling supports the following functionality for FMU participants:

Table 28: Supported capabilities for FMU participants

Analysis types	• Steady • Transient
Coupling participant-type combinations	• Ansys-Ansys • Ansys-FMU • FMU-FMU
Quantities that can be sent to or received by an FMU participant	• Temperature • Heat rate • Heat Transfer Coefficient + Convection Reference Temperature[1] (p. 359)
Data transfer topologies[2] (p. 359)	• Undefined⇒ – Undefined

	– Surface – Volume – Planar surface • Undefined⇐ – Undefined – Surface – Volume – Planar surface
1: Both quantities must be defined for this thermal data transfer to be valid. **2:** Currently, only transfers of single-valued scalar data are supported.	

Co-Simulation with an FMU Participant

For information on setting up a co-simulation that includes an FMU participant, see:

Adding an FMU Participant

Adding an FMU Proxy Participant

Adding Coupling Interfaces and Data Transfers

Adding an FMU Participant

To add participant definitions (p. 121), use the **Add Participant** dialog or run the `AddParticipant()` command for each participant. For FMU participants, set the **Input File** field or the `InputFile` argument to the `*.fmu` file.

Note:

You can use the same `*.fmu` file multiple times. For example, you might load three instances of it — that is, adding three distinct participants defined by that file.

When the dialog is used or the command is run, the participant definitions are loaded and the data model is populated with information from the `*.fmu` file.

Example 55: Add an Ansys participant and an FMU participant in the CLI

```
>>> AddParticipant(InputFile = 'cfx.scp')
 >> AddParticipant(InputFile = 'spring.fmu')
```

Example 56: Add one Ansys participant and two FMU participants defined by the same file in the CLI

```
>>> AddParticipant(InputFile = 'cfx.scp')
>>> AddParticipant(InputFile = 'spring.fmu')
>>> AddParticipant(InputFile = 'spring.fmu')
```

Adding an FMU Proxy Participant

An FMU proxy participant is alternative method for running FMUs. Instead of being run within the System Coupling environment, they run in a separate process that has a separate environment. It allows you to run FMU files that are not capable of running in the current System Coupling co-simulation environment.

This participant type must be selected manually, so use it when the default FMU participant fails. For example, if the FMU uses a Python interpreter it may not run with the default method.

Note:

- Not all FMU files can be run with this method as the proxy participant uses a subset of the FMI standard.
- This method runs significantly slower than a default FMU participant.
- The FMU proxy participant does not support:
  - Non-real variables: integer, boolean, string, enumeration
  - Parameters
  - Initial variable values
  - Variable bounds

To add participant definitions (p. 121), use the **Add Participant** dialog or run the `AddParticipant()` command for each participant. For FMU Proxy participants:

- Set the **Input File** field or the `InputFile` argument to the `*.fmu` file.
- Set the **Participant Type** field or the `ParticipantType` argument to **`FMU-PROXY`**.

Note:

You can use the same `*.fmu` file multiple times. For example, you might load three instances of it — that is, adding three distinct participants defined by that file.

When the dialog is used or the command is run, the participant definitions are loaded and the data model is populated with information from the `*.fmu` file.

Example 57: Add an Ansys participant and an FMU Proxy participant in the CLI

```
>>> AddParticipant(InputFile = 'cfx.scp')
 >> AddParticipant(InputFile = 'spring.fmu', ParticipantType = 'FMU-PROXY')
```

Adding Coupling Interfaces and Data Transfers

When a coupling interface and data transfers are added to the co-simulation, an FMU participant's information is populated to the data model from the participant's `.fmu` file.

Adding a Coupling Interface

To add a coupling interface (p. 124) that includes an FMU participant, select one of the following options:

- Use the **Add Coupling Interface** context menu option.
- Run the `AddInterface()` command, using internal object names as arguments to specify the participants be included in the interface.

Note:

For information on naming conventions and restrictions for participant objects, see FMU `CouplingParticipant` Data Model Settings (p. 362)..

Adding Data Transfers

To add a data transfer (p. 126) that includes an FMU participant, select one of the following options:

- Use the **Add Data Transfer** context menu option.
- Use the **Add Ordered Data Transfers** context menu option to add all possible data transfers between two FMUs.
- Run the AddDataTransfer() command.
- Run the AddOrderedDataTransfers() command.

Data transfers can only occur when the source and target variables are of the same data type (that is, real-to-real, integer-to-integer, etc.). Given that Ansys participants can send/receive real-valued variables, only variables with a `DataType` of ***`Real`*** can be sent in an FMU-to-Ansys participant transfer.

Note:

For information on naming conventions and restrictions for variable and parameter objects, see `Variable` (object) (p. 365) and `FMUParameter` (object) (p. 368), respectively.

Example 58: Add a coupling interface and a data transfer with an Ansys participant and an FMU participant in the CLI

```
>>> interface = AddInterface(SideOneParticipantName = "FLUENT-1",
      SideOneRegionNames = ["Volume5"],
      SideTwoParticipantName = "FMU-2",
      )

>>> AddDataTransfer(Interface = interface,
      TargetSide = "One",
      SideOneVariable = "Temperature",
      SideTwoVariable = "Real_0"
      )
```

Example 59: Add a coupling interface and data transfers with two FMU participants in the CLI

```
 >>> interface = AddInterface(SideOneParticipantName = "FMU-1",
      SideTwoParticipantName = "FMU-2",
      )

>>> AddOrderedDataTransfers(Interface = "interface")
       ['Real_1', 'Real_2', 'Real_3']
```

FMU `CouplingParticipant` Data Model Settings

Contains settings that define the FMU participant.

FMU participants are named according to the same convention that is used for Ansys participants: `<ParticipantType (p. 363)>-#`, where:

"*ParticipantType*" is the type of participant, and

"*#*" is an index (starting with ***1***) indicating the order in which the participant was loaded.

Note:

FMU and Ansys participants are indexed in the same sequence.

Coupling Participant object contents:

DisplayName (setting)

ParticipantType (setting)

ParticipantFileLoaded (setting)

LoggingOn (setting)

CanSerializeFmuState (setting)

CanGetAndSetFmuState (setting)

DisplayName (setting)

Participant name to be shown in user-facing communications.

ParticipantType (setting)

Indicates the type of coupling participant. For FMU participants, the only possible value is *FMU*.

ParticipantFileLoaded (setting)

Indicates the `.fmu` file used to add the participant. If this file is located in the coupling working directory, only the filename is needed. Otherwise, the path to the file's location must be provided.

LoggingOn (setting)

Indicates whether debug logging is enabled for the FMU participant.

Possible values:

- ***False***: Disabled (default value)

  A transcript tab is not displayed in the System Coupling GUI for the participant.

- ***True***: Enabled

  A transcript tab is displayed in the System Coupling GUI for the participant.

  A log file with the base name of the FMU is written to the root level of the analysis. For example, if the FMU is named `myFMU.fmu`, the log file is named `myFMU.log`. The first line of the file is a timestamp in the format of date, time, then time zone. For example:

  ```
  9/8/2023 2:57:33 PM Eastern Daylight Time
  ```

  The timestamp indicates when the log file was started.

CanSerializeFmuState (setting)

Indicates whether restart information is written for the FMU (during a restart step).

Possible values:

- ***False***: Disabled (default value)
- ***True***: Enabled

It will only be True if `CanSerializeFMUstate` is True in the FMU setup. In the FMI standard, this also means that `CanGetAndSetFMUstate` is also True.

`CanGetAndSetFmuState` (setting)

Indicates whether the FMU can get and set its state.

Possible values:

- ***`False`***: Disabled (default value)
- ***`True`***: Enabled

  If `CanSerializeFMUstate` is True in the FMU setup, this also means that `CanGetAndSet-FMUstate` is also True.

Note:

FMUs can only perform iterations if `canGetAndSetFMUstate` is True, since FMUs are required to rewind to the beginning of the step for every iteration. To do this, the initial state for the step must be saved (prior to the start of the first iteration) and that state set in the FMU at the beginning of each iteration. If the option is False, then this is not possible.

If the value is set to False, the following validation warning is displayed:

```
Cosimulation is defined with multiple iterations but participant
ParticipantName does not support multiple iterations. This
participant will only update its variables during the first
coupling iteration of each step.
```

`Input Variables` (setting)

Specifies the variables for which the participant can receive data.

Blank values are permitted.

`OutputVariables` (setting)

Specifies the variables for which the participant can send data.

Blank values are permitted.

UpdateControl (singleton)

Settings defined under the UpdateControl singleton control the frequency with which the participant performs updates during the execution of a coupled analysis.

Note:

Every participant performs an update during the first iteration of each run (that is, during the first step of both initial and restarted coupling runs), regardless its UpdateControl settings.

UpdateControl singleton contents:

Option (setting)

Specifies how often the participant will perform updates.

Possible values:

- ***ProgramControlled*** (default value)

  The FMU participant performs an update for every coupling iteration of each coupling step (default value).

  However, if the FMU participant cannot get and set its internal state, it updates only for the first coupling iteration of each coupling step.

- ***EveryIteration***

  The FMU participant updates for every coupling iteration of each coupling step. This is the same behavior as ***ProgramControlled*** when the FMU can get and set its internal state.

Variable (object)

Contains settings that define the details for each of the variables for which the participant can send or receive data.

FMU variables are named according to a naming convention based on the variable data type: <*Data-Type (p. ?)*>_#, where "#" is an integer-valued reference value used internally by the FMU to identify the variable. Variable names therefore must not be changed by the user, as the name of the variable itself is used to identify it within the FMU.

Variable object contents:

DisplayName (setting)

Variable name to be displayed in user-facing communications.

Optional. If not specified, the variable object name is used.

`DataType` (setting)

Data type of the variable being transferred.

Required.

Possible values:

- ***`Real`***
- ***`Integer`***
- ***`Logical`***
- ***`String`***
- ***`Enumeration`***

Settings available when the variable's DataType is set to Real:

`RealInitialValue` (setting)

Starting value of the variable's real data.

Accepts real values. Default value is *`0.0`*.

`RealMin` (setting)

Minimum allowable value of the variable's real data.

Accepts real values lesser than or equal to the `RealMax` value.

Optional. Blank values are permitted. If no value is provided, a minimum boundary is not enforced.

`RealMax` (setting)

Maximum allowable value of the variable's real data.

Accepts real values greater than or equal to the `RealMin` value.

Optional. Blank values are permitted. If no value is provided, a maximum boundary is not enforced.

Settings available when the variable's DataType is set to Integer:

`IntegerInitialValue` (setting)

Starting value of the variable's integer data.

Accepts integer values. Default value is *`0`*.

`IntegerMin` (setting)

Minimum allowable value of the variable's integer data.

Accepts integer values lesser than or equal to the `IntegerMax` value.

Optional. Blank values are permitted. If no value is provided, a minimum boundary is not enforced.

IntegerMax (setting)

Maximum allowable value of the variable's integer data.

Accepts integer values greater than or equal to the IntegerMin value.

Optional. Blank values are permitted. If no value is provided, a maximum boundary is not enforced.

Settings available when the variable's DataType is set to Logical:

LogicalInitialValue (setting)

Starting value of the parameter's logical data.

Accepts logical (Boolean) values.

Possible values:

- ***True*** (default value)
- ***False***

Settings available when the variable's DataType is set to String:

StringInitialValue (setting)

Starting value of the variable's string data.

Accepts string values. Default value is ***None***.

Settings available when the variable's DataType is set to Enumeration:

EnumerationInitialValue (setting)

Starting value of the variable's enumeration data.

Accepts integer values. Default value is *0*.

EnumerationMin (setting)

Minimum allowable value of the variable's enumeration data.

Accepts integer values lesser than or equal to the EnumerationMax value.

Optional. Blank values are permitted. If no value is provided, a minimum boundary is not enforced.

EnumerationMax (setting)

Maximum allowable value of the variable's enumeration data.

Accepts integer values greater than or equal to the EnumerationMin value.

Optional. Blank values are permitted. If no value is provided, a maximum boundary is not enforced.

`FMUParameter` (object)

Contains settings that define the details for each of the parameters for which the participant can send or receive data.

FMU variables are named according to a naming convention based on the parameter data type: <*`DataType (p. ?)`*>_#, where "#" is an integer-valued reference value used internally by the FMU to identify the parameter. Parameter names therefore must not be changed by the user, as the name of the parameter itself is used to identify it within the FMU.

Unlike variables, FMU parameters are typically not changed during the co-simulation. FMU parameters are not included in the interface and so cannot be used as connections, inputs, outputs, etc.

`FMUParameter` object contents:

`DisplayName` (setting)

Parameter name to be displayed in user-facing communications.

Optional. If not specified, the parameter object name is used.

`DataType` (setting)

Data type of the parameter being transferred.

Required.

Possible values:

- ***`Real`***
- ***`Integer`***
- ***`Logical`***
- ***`String`***
- ***`Enumeration`***

Settings available when the parameter's DataType is set to Real:

`RealValue` (setting)

Starting value of the parameter's real data.

Accepts real values. Default value is ***`0.0`***.

`RealMin` (setting)

Minimum allowable value of the parameter's real data.

Accepts real values lesser than or equal to the RealMax value.

Optional. Blank values are permitted. If no value is provided, a minimum boundary is not enforced.

RealMax (setting)

Maximum allowable value of the parameter's real data.

Accepts real values greater than or equal to the RealMin value.

Optional. Blank values are permitted. If no value is provided, a maximum boundary is not enforced.

Settings available when the parameter's DataType is set to Integer:

IntegerValue (setting)

Starting value of the parameter's integer data.

Accepts integer values. Default value is ***0***.

IntegerMin (setting)

Minimum allowable value of the parameter's integer data.

Accepts integer values lesser than or equal to the IntegerMax value.

Optional. Blank values are permitted. If no value is provided, a minimum boundary is not enforced.

IntegerMax (setting)

Maximum allowable value of the parameter's integer data.

Accepts integer values greater than or equal to the IntegerMin value.

Optional. Blank values are permitted. If no value is provided, a maximum boundary is not enforced.

Settings available when the parameter's DataType is set to Logical:

LogicalValue (setting)

Starting value of the parameter's logical data.

Accepts logical (Boolean) values.

Possible values:

- ***True*** (default value)
- ***False***

Settings available when the parameter's DataType is set to String:

StringValue (setting)

Starting value of the parameter's string data.

Accepts string values. Default value is ***None***.

Settings available when the parameter's DataType is set to Enumeration:

`EnumerationInitialValue` (setting)

Starting value of the parameter's enumeration data.

Accepts integer values. Default value is *0*.

`EnumerationMin` (setting)

Minimum allowable value of the parameter's enumeration data.

Accepts integer values lesser than or equal to the `EnumerationMax` value.

Optional. Blank values are permitted. If no value is provided, a minimum boundary is not enforced.

`EnumerationMax` (setting)

Maximum allowable value of the parameter's enumeration data.

Accepts integer values greater than or equal to the `EnumerationMin` value.

Optional. Blank values are permitted. If no value is provided, a maximum boundary is not enforced.

Common Engineering Applications for System Coupling

System Coupling provides coupling management for a wide range of engineering applications. For more information, see:

Fluid-Structure Interaction (FSI)

Aerodynamic Damping Analysis

Electric Arc Modeling

Fluid-Structure Interaction (FSI)

This section provides best practices information and recommendations for working with Fluid-Structure Interaction (FSI) problems. For more information, see:

Building a Coupled FSI Analysis from Decoupled Participant Problems

Choosing the Mesh-Related Timescale

Defining Mesh Motion and Deformation Settings

Note:

For participant-specific details on FSI problems, see the corresponding product documentation:

- FSI Setup Recommendations for Fluent-Mechanical Couplings in the *Fluent User's Guide*
- Fluid Structure Interaction in the *CFX Reference Guide*
- Fluid-Structure Interaction (FSI) in the *Mechanical User's Guide*

Building a Coupled FSI Analysis from Decoupled Participant Problems

If it will be advantageous to run an FSI problem as a coupled analysis, then the recommended approach is to build up to the final analysis in stages of progressive complexity. This section takes the generalized process described in Building Up to a Coupled Analysis (p. 393) and applies it to the incremental creation of a two-way coupled FSI analysis.

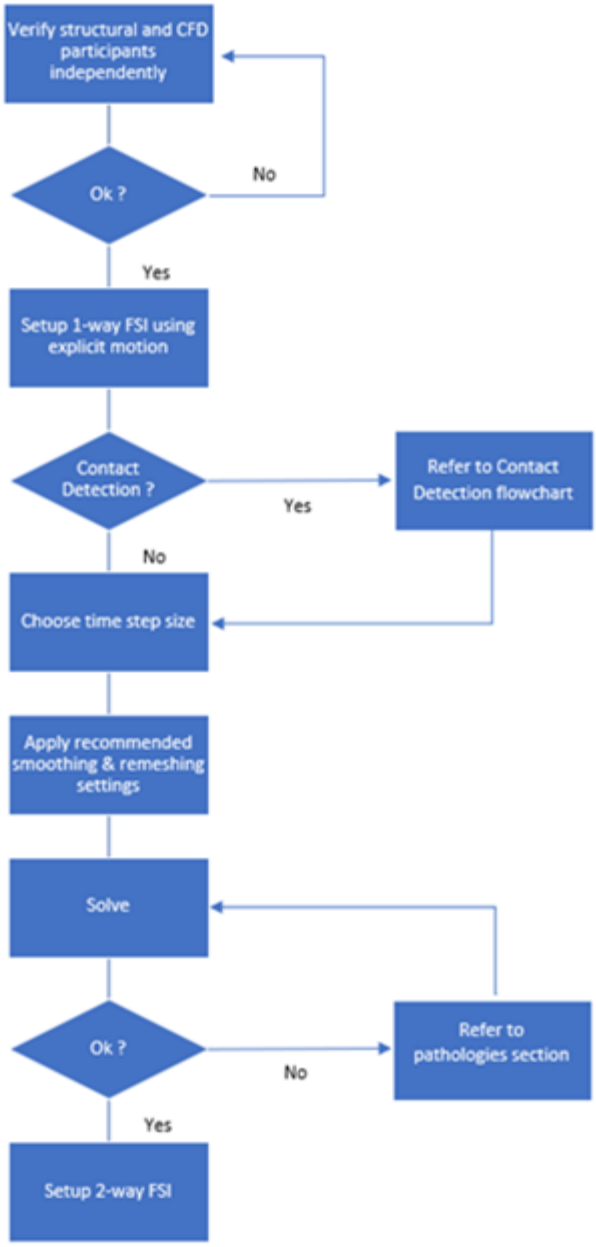
Step 1: Solve the Fluid and Structural Problems Independently

Step 2: Solve the Fluid and Structural Problems as One-Way Coupled Analyses

Step 3: Solve the Fluid and Structural Problems as a Two-Way Coupled Analysis

The procedure for building a coupled analysis is summarized below in Figure 75: Workflow to set up a coupled FSI analysis (p. 372).

Figure 75: Workflow to set up a coupled FSI analysis



Note:

In the following sections, the structural solver is assumed to be Ansys Mechanical. The fluid solver may be any Ansys CFD solver that supports System Coupling FSI simulations.

Step 1: Solve the Fluid and Structural Problems Independently

Because the constituent fluid and structural problems may each be quite complex on their own, ensure that they can each be solved individually before attempting to join them in a coupled analysis. Solve the problems independently, as follows:

Fluid analysis:

Solve only for the fluid-flow part of the problem (that is, do not solve for mesh motion).

To set up the fluid-flow analysis:

1. Remove any System Coupling motion specifications that would be used to verify mesh motion and remeshing in the fluid analysis.
2. Introduce the initial and boundary conditions.
3. Define a motion (structural displacement) as an input. This should approximate the motion (or range of motion) expected from the structural problem.

Structural analysis:

Solve only for the explicit motion and preliminary contact (with no offset) part of the problem (that is, do not solve for fluid force).

To set up the structural analysis:

1. Remove any System Coupling motion specifications (if they exist) that would be used to verify mesh motion and remeshing in the fluid analysis.
2. Introduce the initial and boundary conditions.
3. If applicable, add contact and contact offsets to ensure that the fluid mesh does not collapse.
4. Define a load (fluid force) as an input. This should approximate the load (magnitude and distribution) expected from the fluid analysis.

Once you have verified that each of the individual problems can be solved successfully on its own, you can incorporate them into a series of one-way coupled analyses (p. 373).

Step 2: Solve the Fluid and Structural Problems as One-Way Coupled Analyses

In this step, incrementally add complexity and non-linearity by setting up and solving each problem as a one-way coupled analysis.

Solve the Fluid Problem as a Series of One-Way Coupled Analyses

Set up and run two one-way coupled analyses with a structural source participant and a fluid target participant, where the displacement data generated by the structural analysis is applied to the fluid analysis. Solve for the motion transfer, first without and then with fluid-flow calculations.

Analysis 1: Motion transfer to fluid (without solving flow equations)

First, solve for the transfer of motion from the structural analysis to the fluid analysis, but without solving the flow equations. This allows you to test mesh motion and remeshing. It can help you to identify and address issues that cause mesh folding, as well as to determine contact offsets and timescales.

To set up the analysis without flow equations:

1. Remove the approximated motion input that was applied in the individual fluid analysis.
2. Add System Coupling mesh motion to the analysis.
3. For the fluid side of the analysis, configure mesh motion and deformation (p. 377) settings. If applicable, set contact detection parameters.
4. For the structural side of the analysis, configure contact offsets to address limitations in the fluid analysis.
5. Generate System Coupling files.
   - If using one of System Coupling user interfaces, generate a System Coupling Participant (`.scp`) file for each participant.
   - If using System Coupling in Workbench, generate a System Coupling Input (`.sci`) file for the analysis.

   Note:

   You can export a Workbench-based coupled analysis for execution in one of System Coupling's user interfaces. For details, see Exporting a System Coupling Setup (p. 273).
6. Complete the coupled analysis setup (duration, coupling steps, coupling iterations, and so on) according to the System Coupling context being used.

When this analysis has been successfully solved, move on to solving the analysis with flow equations.

Analysis 2: Motion transfer to fluid (with solving flow equations)

Next, expand on the previous one-way analysis, this time activating the flow equations so that they will be solved with the transfer of motion from structural analysis. This allows you to verify that the flow equations can still be solved after the introduction of motion to the analysis.

To set up the analysis with flow equations:

1. Remove the results from the individual structural analysis.
2. Activate flow equations for the fluid analysis.
3. Regenerate the System Coupling files to capture the changes to the participant physics, as described previously for Analysis 1.

Once the one-way fluid analysis has been solved with flow equations, how you proceed depends on your end goal for the problem:

- If a one-way coupled analysis is your end goal, then the next step is to evaluate its convergence. For information on assessing coupled analysis convergence, see Evaluating Convergence

and Data Transfer Accuracy (p. 397). To evaluate participant convergence, use the method provided by the CFD solver.

- If a two-way coupled analysis is your end goal (or if you observe significant deformation of the fluid domain because of the structural motion), then proceed to the creation of the two-way analysis.

> **Note:**
>
> Before doing so, ensure that you have also completed the intermediate step of solving the structural problem as a one-way coupled analysis (p. 375).

Solve the Structural Problem as a One-Way Coupled Analysis

Solve for the transfer of Force from the fluid analysis to the structural analysis as a one-way coupled analysis by adding coupling specifications to the problem. This allows you to verify that the structural equations can still be solved after the introduction of applied force to the analysis.

Additionally, in setting up and running the one-way analysis, you may be able to identify and address potential issues regarding timescales, mesh folding, maximum stress values, and special solution considerations (for example, mixed u-P formulation for nearly incompressible hyperelastic solids).

Force transfer to structural

When setting up the one-way structural analysis:

1. Remove the approximated load that was applied in the individual structural analysis.
2. Add a **System Coupling Region** to the analysis.
3. Suppress the displacement data transfer so that the analysis is one-way.
4. Generate System Coupling files.
   - If using one of System Coupling user interfaces, generate a System Coupling Participant (`.scp`) file for each participant.
   - If using System Coupling in Workbench, generate a System Coupling Input (`.sci`) file for the analysis.

   > **Note:**
   >
   > You can export a Workbench-based coupled analysis for execution in one of System Coupling's user interfaces. For details, see Exporting a System Coupling Setup (p. 273).

5. Complete the coupled analysis setup (duration, coupling steps, coupling iterations, and so on) according to the System Coupling context being used.

Once the one-way structural analysis has been solved, how you proceed depends on your end goal for the problem:

- If a one-way coupled analysis is your end goal, then the next step is to evaluate its convergence. For details on assessing coupled analysis convergence, see Evaluating Convergence and Data Transfer Accuracy (p. 397). To evaluate participant convergence, use the method provided by the structural solver.
- If a two-way coupled analysis is your end goal (or if you observe a significant structural deformation because of the applied load), then proceed to the creation of the two-way analysis.

Note:

Before doing so, ensure that you have also completed the intermediate step of solving the fluid problem as a one-way coupled analysis (p. 373).

Step 3: Solve the Fluid and Structural Problems as a Two-Way Coupled Analysis

Solve the fluid and structural problems as a two-way coupled analysis.

When setting up the two-way simulation:

1. For the fluid analysis, remove the initial structural displacement input that was applied to approximate the expected motion. This allows the actual displacement data generated by the structural analysis to be calculated and applied to the fluid analysis.
2. For the structural analysis, remove the initial fluid-force load (explicit displacement) that was applied to approximate the expected load. This allows the force data generated by the fluid to be calculated (once the transport equations are solved) and applied to the structural analysis.
3. Incorporate each participant problem into the two-way simulation.

Choosing the Mesh-Related Timescale

Choosing the correct time step size for System Coupling is critical, since an incorrect time step size can lead to mesh folding and other potential issues. Mesh deformation is one aspect of an appropriate participant timescale and should always be evaluated against other relevant timescales in each participant.

To determine the mesh deformation time scale, calculate the time step Δt size based on:

- The length of the smallest cell, Δs
- The maximum expected velocity of the mesh, v

$$\Delta t = \frac{\Delta s}{\Delta v}$$

The maximum expected velocity of the mesh can be determined through Mechanical by examining the total deformation occurring due to the explicit motion. The maximum total deformation can be divided by the time step size to estimate the expected mesh velocity of the CFD mesh.

As a starting point, attempt to define the boundary so that it can move only a single cell in each time step. Then, once a successful run has been established, the time step size can be increased.

For participant-specific details, see the corresponding product documentation:

- Performing Time-Dependent Calculations in the *Fluent User's Guide*
- Timestep Selection in the *CFX-Solver Modeling Guide*
- Steps and Step Controls for Static and Transient Analyses in the *Mechanical User's Guide*

Defining Mesh Motion and Deformation Settings

For participant-specific recommendations on mesh motion and deformation settings, see the corresponding CFD product documentation:

CFX:

- Mesh Motion Options in the *CFX-Solver Modeling Guide*
- Mesh Deformation in the *CFX-Pre User's Guide*

Fluent:

- Using Dynamic Meshes in the *Fluent User's Guide*
- Specifying the Motion of Dynamic Zones in the *Fluent User's Guide*

Aerodynamic Damping Analysis

System Coupling provides an easy and systematic workflow to transfer mode shapes from a Mechanical modal analysis to a CFD analysis for Aerodynamic Damping analysis (also known as "Aerodamping" or "Blade Flutter" analysis).

System Coupling maps the mode shapes from the Mechanical modal analysis surface mesh to a CFD surface mesh. It provides geometry alignment, interface instancing, unmapped value handling, as well as other functionality. The mapped mode shapes are written to `.csv` file(s) that can be used to provide displacements for a CFD analysis.

For participant-specific details on aerodamping problems, see the corresponding product documentation:

- Modal Analysis in the *Mechanical User's Guide*
- Modal Analysis in the Mechanical APDL *Structural Analysis Guide*
- Blade Flutter in the *CFX-Solver Modeling Guide*

For more information on coupled aerodamping analyses, see:

Coupling Participants

Coupling Interfaces

Handling Participant Setup Changes

Coupling Participants

System Coupling supports two specialized server participants that are used for aerodamping co-simulations:

- **Mechanical Server**
  - `ParticipantType` = ***`MECH-SRV`***
- **CFD Server**
  - `ParticipantType` = ***`CFD-SRV`***

Verify Coupling Participant Input Files

Verify that the necessary input file has been generated for each coupling participant. You will use these files to add the participants to the aerodamping co-simulation.

Mechanical Server

- **Input Files:** Mechanical modal analysis results file (`.rst`).

  Note:

  If modal mass is defined, a `.mode` file is also used.

- For information on how to perform the modal analysis and to generate this file, see Modal Analysis in the *Mechanical User's Guide*.

CFD Server

- **Input File:** CFD surface mesh file (`.csv`)
- For information on how to create this file, see Establishing Profile Data and Creating Profile Functions in the *CFX-Pre User's Guide*.

Add Coupling Participants

Once you have the participant input files, you may add the participants to the aerodamping co-simulation using either System Coupling's GUI or CLI.

Add a participant in the GUI:

1. Right-click the **Setup** branch and select **Add Participant**.

2. In the **Open** dialog, select the participant's input file (`.rst` or `.csv`).

Add a participant in the CLI:

Run the `AddParticipant()` command, as shown in the following examples:

- Add the Mechanical Server participant using the `file.rst` results file located in Mechanical's **modal** directory:

```
AddParticipant(ParticipantType = 'MECH-SRV', InputFile = os.path.join('modal', 'file.rst'))
```

- Add the CFD Server participant using the `input.csv` file located in CFX's **cfd** directory:

```
AddParticipant(ParticipantType = 'CFD-SRV', InputFile = os.path.join('cfd, 'input.csv'))
```

Define the Mechanical Server

The Mechanical Server participant defines one or more surface regions that are available to transfer the data. Since this participant is used only as the source for mapping the data to the CFD Server, it will define only the *output* variables.

Note:

In the **System Coupling** GUI, the **Input Variables** setting is disabled for regions and may be disregarded.

Variable Names

By default, Mechanical Server's mode shape `Variable` objects are named according to the following conventions:

- For modal analyses *with* cyclic symmetry: `ModeShape_Mode<`*N*`>_HI<`*M*`>`, where *N* is the mode number and *M* is the harmonic index
- For modal analyses *without* cyclic symmetry: `ModeShape_Mode<`*N*`>`, where *N* is the mode number

You may change the variable's `DisplayName`, if you wish.

Variable Properties

Each mode shape variable is described by the following properties:

- `DataType`: ***`Real`*** or ***`Complex`***. If the modal analysis model features cyclic symmetry, the base and duplicate pair mode shapes form the real and imaginary components of complex mode shapes.
- `QuantityType`: ***`Mode Shape`***

- `TensorType`: ***`Vector`***
- `IsExtensive`: ***`False`*** (meaning that the variable is intensive)

Variable Attributes

Each mode shape variable is further described by specialized attributes defined as `Attribute` objects:

- `Frequency`: The frequency for this mode shape (in Hz).
- `MaxDisplacement`: The maximum displacement value for the entire mode. Note that Mechanical Server normalizes each mode shape by the maximum displacement value.
- `ModeNumber`: The mode number of a given mode shape variable (integer).
- `Mass`: The mass value for the entire model.

Define the CFD Server

The CFD Server participant typically defines the surface regions that can receive mode shape values from the Mechanical Server participant. Note that input variables are not usually defined by the participant. Instead, you will typically define the variables to be received when you set up the coupling interface in System Coupling.

Note:

In the **System Coupling** GUI, the **Input Variables** and **Output Variables** settings are disabled for regions and may be disregarded.

Optionally, as part of your setup, you may also:

Set Execution Control Parameters

Define Participant Instancing

Set Execution Control Parameters

CFD Server has several specialized parameters in its `ExecutionControl` object. You my use these parameters to customize its behavior:

- `BaseOutputFileName`

  The base output file name that is used to create the output file.

  By default, the value of this parameter is ***`output`***, which means that the resulting output files written by CFD Server will be named according to the convention `output_<`*`variable name`*`>.csv`.

- `OverwriteExistingFiles`

  Boolean flag that tells CFD Server how it should behave if the file already exists:

  – ***`False`***: Issue an error and do not overwrite the file. (default value)

– ***True***: Overwrite the file without issuing an error.

Define Participant Instancing

You can use System Coupling's interface instancing capabilities to apply instancing to a CFD Server participant.

To define participant instancing for CFD Server, perform the following steps:

1. In System Coupling's `Library`, create an Instancing object, as described in Adding Instancing Objects (p. 156).

2. Define an instancing object on CFD Server using its `CouplingParticipant.Instancing` setting, as follows:

   If using the GUI:

   1. In the outline, select the CFD Server participant.

   2. Under properties, select an instancing object from the **Instancing** setting's drop-down menu.

   If using the CLI:

   Define an instancing object on the participant, as shown below:

   ```
   DatamodelRoot().CouplingParticipant['CFD-SRV]'.Instancing = 'Instance-1'
   ```

   If no value is provided for this setting, then it defaults to *None* (no instancing is defined).

Once participant instancing is defined, CFD-Server handles all aspects of the instancing and provides relevant information as arguments for its execution command. When the case is run, CFD Server provides a mesh that contains all instances.

Participant instancing objects have the same options, behaviors, and validations as interface instancing objects. For details, see Defining Interface Instancing for Cylindrical Geometry Models (p. 155) in this guide and `Instancing` in the *System Coupling Settings & Commands Reference* documentation.

Coupling Interfaces

To perform a co-simulation, you must add a coupling interface to the analysis. For details, see:

Add a Coupling Interface

Add Data Transfers

Using Expressions

Add a Coupling Interface

To transfer the mode shapes from Mechanical Server to CFD server, you must define a coupling interface. You may add coupling interfaces using either System Coupling's GUI or CLI.

Add a coupling interface in the GUI

1. Right-click the **Setup** branch and select **Add Coupling Interface**.

   The coupling interface object is added to the data model under the **Coupling Interface** branch.

2. Verify that the two sides of the interface are defined correctly and adjust as needed.

Add a coupling interface in the CLI

Run the `AddInterface()` command, as in the following example:

```
AddInterface(
    SideOneParticipant = 'MECH-SRV-1', SideOneRegions = ['BLADESURF'],
    SideTwoParticipant = 'CFD-SRV-2', SideTwoRegions = ['R1 Blade'])
```

Add Data Transfers

Data transfers define which variables are transferred from one side of the coupling interface to the other side. For aerodamping, these are the mode shape variables. You may add data transfers using either System Coupling's GUI or CLI.

Add a data transfer in the GUI

In the GUI, you may add multiple data transfers at the same time.

1. Right-click the coupling interface object and select **Add Aerodamping Data Transfers**.

   The **Select Mode Shapes** dialog opens.

2. Select the mode shape variables to be transferred (press **Shift + Ctrl** keys to multiselect).

3. Click **Apply**.

A data transfer is added for each of the mode shape variables selected.

Add a data transfer in the CLI

In the CLI, you must add each data transfer individually.

Run the `AddDataTransfer()` command, as in the following example:

```
AddDataTransfer(
    Interface = interfaceName,
    TargetSide = 'Two',
    SourceVariable = 'ModeShape_Mode1',
    TargetVariable = 'ModeShape_Mode1')
```

Using Expressions

Expressions are supported for transfers of real-valued mode shapes. You may apply expressions using either System Coupling's GUI or CLI.

Apply an expression in the GUI

1. Verify that **Data Transfer | Option** is set to ***Using Expression***.
2. Enter the expression for the **Value** setting.

Apply an expression in the CLI

1. Verify that `DataTransfer.Option` is set to ***UsingExpression***.

```
DatamodelRoot().CouplingInterface['Interface-1'].DataTransfer['ModeShape_Mode_1_Index_0'].Option
UsingExpression
```

2. Set `Value` to the expression value.

```
DatamodelRoot().CouplingInterface['Interface-1'].DataTransfer['ModeShape_Mode_1_Index_0'].Value='tempe
```

For more information, see Expressions in System Coupling (p. 59).

Handling Participant Setup Changes

Changes to a server participant's setup after the participant has been added to the co-simulation may cause errors. Although this is an uncommon occurrence, System Coupling minimizes the risk of such errors by checking the server input files against the initial input files. If System Coupling detects changes, it blocks mapping until the participant is updated.

If a participant input file has been updated, then System Coupling writes an error message (either to the **Messages** tab for the GUI or to the CLI console when you attempt to map the analysis) and blocks mapping until the participant is updated. To avoid this error, you can check for input changes before initiating a solve.

If an input file has been updated and no variables or regions have been removed from the original setup, you can update the participant.

Note:

- Cases created before the 2025 R1 release could have input changes even if the `HasInputFileChanged()` command returns a false value. Therefore, Ansys recommends that you update the Mechanical server participant for all cases created before the 2025 R1 release.

- If the participant can't be updated, then you must delete the participant from the analysis and then re-add it using the updated input file.

Handling participant setup changes in the GUI

The **System Coupling** GUI checks for participant setup changes automatically. If the **Messages** tab indicates that a participant input has been updated, update the participant by performing the following steps:

1. Right-click the server participant with the changed input file.
2. Select **Update participant**.
3. Right-click the **Coupling Participants** branch, select **Add participant**, and re-add the participant using the updated input file.

Handling participant setup changes in the CLI

The System Coupling CLI checks for participant setup changes at runtime. Before starting a solve, you may also check for changes manually and update the participant if needed. To do so, perform the following steps:

1. You may check for changes to the input file of a participant by running the `HasInputFileChanged()` command, adding the participant's name as an argument.

```
>>> HasInputFileChanged(ParticipantName="<participant_name>")
True
```

   If the command returns a value of *`False`*, then you may start the solve. If it returns a value of *`True`*, then update the participant before proceeding.

2. Update the participant by running the `UpdateParticipant()` command with the participant's name as an argument. Optionally, you may also specify the name of the input file.

```
>>> UpdateParticipant(
     ParticipantName="<participant_name>",
     InputFile="<input_file>"
    )
```

Performing Mapping

The mapping process involves mapping the modal shapes from Mechanical Server to CFD Server. You may perform mapping using either 's GUI or CLI. If you are using System Coupling's GUI, you may preview the mapping before starting the mapping process.

Note:

If a participant's input file has been changed, you need to take corrective action to continue the co-simulation with the modified setup. For details, see Handling Participant Setup Changes (p. 383).

When you preview the mapping, the results are not written to a `.csv` output file, and the mapping results open in EnSight automatically. When you perform the mapping, the results are written to the `.csv` file.

Preview mapping in the GUI

Right-click the **Mapping** branch and select **Preview Mapping**.

Perform mapping in the GUI

Right-click the **Mapping** branch and select **Map**.

Perform mapping in the CLI

Run the `Solve()` command.

```
Solve()
```

Verifying Mapping Results

Use the following methods to check your mapping results:

Review the System Coupling Transcript

Visualize Mapped Results in EnSight

Review the System Coupling Transcript

During the mapping process, System Coupling writes a Transcript which logs relevant steps in the mapping process. The following sections are of particular interest for aerodamping applications:

Mesh Statistics

Mapping Summary and Transfer Diagnostics

For more detailed information about the System Coupling Transcript, see Transcript and Log File (p. 332). (Note that only the sections relevant to aerodamping analyses will be included in the Transcript.)

Mesh Statistics

Use the Mapping Summary (p. 342) section of the Transcript to verify that the source and target meshes are as expected.

```
+-------------------------------------------------------------------------+
|                            MESH STATISTICS                              |
+-------------------------------------------------------------------------+
| Participant: CFD-SRV-2                                                  |
|   Number of face regions                                            1 |
|     Number of faces                                            26 824 |
|       Quadrilateral                                            26 824 |
|     Area (m2)                                                6.709e-02 |
|   Bounding Box (m)                                                      |
|     Minimum                          [ 1.336e-01 -2.107e-01 -9.922e-02] |
|     Maximum                          [ 3.123e-01  6.076e-02  1.866e-18] |
|                                                                         |
| Participant: MECH-SRV-1                                                 |
|   Number of face regions                                            1 |
|     Number of faces                                             5 658 |
|       Triangular6                                               2 717 |
|       Quadrilateral8                                            2 941 |
|     Area (m2)                                                3.172e-02 |
|   Bounding Box (m)                                                      |
|     Minimum                          [ 1.426e-01 -2.046e-01 -1.000e-01] |
|     Maximum                          [ 2.784e-01  1.476e-02  3.539e-15] |
|                                                                         |
| Total                                                                   |
|   Number of cells                                                   0 |
|   Number of faces                                              32 482 |
|   Number of nodes                                              42 976 |
+-------------------------------------------------------------------------+
```

Mapping Summary and Transfer Diagnostics

Use the Mapping Summary (p. 342) and Transfer Diagnostics (p. 344) sections of the Transcript to perform a high-level assessment of the mapping quality.

For best results, the target values should be as close to 100% mapped as possible. The source and target values should be similar, although slight differences may occur when the meshes are dissimilar.

```
+-------------------------------------------------------------------------+
|                            MAPPING SUMMARY                              |
+-------------------------------------------------------------------------+
|                                   |      Source            Target       |
+-------------------------------------------------------------------------+
| Interface-1                       |                                     |
|   ModeShape_Mode1                 |                                     |
|     Mapped Area [%]               |       100               >99         |
|     Mapped Elements [%]           |       100               >99         |
|     Mapped Nodes [%]              |       100               >99         |
|   ModeShape_Mode2                 |                                     |
|     Mapped Area [%]               |       100               >99         |
|     Mapped Elements [%]           |       100               >99         |
|     Mapped Nodes [%]              |       100               >99         |
|   ModeShape_Mode3                 |                                     |
|     Mapped Area [%]               |       100               >99         |
|     Mapped Elements [%]           |       100               >99         |
|     Mapped Nodes [%]              |       100               >99         |
|   ModeShape_Mode4                 |                                     |
```

```
|     Mapped Area [%]                |       100              >99           |
|     Mapped Elements [%]            |       100              >99           |
|     Mapped Nodes [%]               |       100              >99           |
|   ModeShape_Mode5                  |                                      |
|     Mapped Area [%]                |       100              >99           |
|     Mapped Elements [%]            |       100              >99           |
|     Mapped Nodes [%]               |       100              >99           |
|   ModeShape_Mode6                  |                                      |
|     Mapped Area [%]                |       100              >99           |
|     Mapped Elements [%]            |       100              >99           |
|     Mapped Nodes [%]               |       100              >99           |
+------------------------------------------------------------------------------+
+------------------------------------------------------------------------------+
|                          Transfer Diagnostics                                |
+------------------------------------------------------------------------------+
| CFD Server                         |                                      |
|   Interface: Interface-1           |                                      |
|     ModeShape_Mode1                |                                      |
|       Weighted Average x           |     2.39E-01          2.39E-01       |
|       Weighted Average y           |    -2.87E-02         -2.88E-02       |
|       Weighted Average z           |    -1.39E-01         -1.39E-01       |
|     ModeShape_Mode2                |                                      |
|       Weighted Average x           |     1.01E-01          1.01E-01       |
|       Weighted Average y           |    -1.55E-02         -1.54E-02       |
|       Weighted Average z           |     7.73E-02          7.78E-02       |
|     ModeShape_Mode3                |                                      |
|       Weighted Average x           |    -9.27E-03         -9.20E-03       |
|       Weighted Average y           |    -1.22E-02         -1.22E-02       |
|       Weighted Average z           |     1.16E-02          1.16E-02       |
|     ModeShape_Mode4                |                                      |
|       Weighted Average x           |     2.72E-02          2.73E-02       |
|       Weighted Average y           |    -5.99E-03         -5.95E-03       |
|       Weighted Average z           |     6.89E-02          6.86E-02       |
|     ModeShape_Mode5                |                                      |
|       Weighted Average x           |    -1.07E-02         -1.08E-02       |
|       Weighted Average y           |     5.57E-03          5.57E-03       |
|       Weighted Average z           |     2.60E-03          2.82E-03       |
|     ModeShape_Mode6                |                                      |
|       Weighted Average x           |    -2.17E-02         -2.19E-02       |
|       Weighted Average y           |    -4.09E-03         -4.01E-03       |
|       Weighted Average z           |     2.23E-02          2.27E-02       |
+------------------------------------------------------------------------------+
```

Visualize Mapped Results in EnSight

Use Ansys EnSight to visualize the mapped results, confirm that the source and target geometries are aligned and instanced properly, and identify areas of non-overlap. You may open mapping results in EnSight from either System Coupling's GUI or CLI.

Open Results in EnSight from the GUI

Right-click the **Solution** branch and select **Open Results in EnSight** from the context menu.

Open Results in EnSight from the CLI

Run the `OpenResultsInEnSight()` command.

```
OpenResultsInEnSight()
```

For more detailed information, see Postprocessing Coupling Results in EnSight (p. 207).

Improving Mapping Results

System Coupling provides several features that you can use to improve the accuracy of mapping for aerodamping.

Geometry Transformations

If the source and target blades do not have consistent orientation in space, you can align the geometries by applying rigid body translation and rotation in the co-simulation setup.

For more information, see Defining Geometry Transformations for Models with Different Orientations (p. 148).

Instancing

In some cases, the number of blade passages for the modal analysis is different from that for the CFD analysis. In such situations, you can use instancing to make the modal analysis and CFD analysis setup consistent.

For more information, see Defining Interface Instancing for Cylindrical Geometry Models (p. 155).

Non-Overlap Handling

Source and target meshes may not overlap perfectly due to missing geometric features on either participant or other inconsistencies. In this case, setting the data transfer's `UnmappedValueOption` setting to ***`Extrapolation`*** may help to fill the unmapped target nodes with physically realistic data profile.

For more information, see Improving Mapping Quality (p. 394) in this guide and `UnmappedValueOption` in the *System Coupling Settings and Commands Reference* documentation.

Performing Remapping

Once you have modified your System Coupling setup to improve mapping results, you can rerun the mapping operation to remap Mechanical Server's modal shapes to CFD Server. You may perform remapping using either System Coupling's GUI or CLI.

Note:

- If a participant's input file has been changed, you need to take corrective action to continue the co-simulation with the modified setup. For details, see Handling Participant Setup Changes (p. 383).
- If results from a previous run exist, the CFD Server participant will raise an error. To perform another mapping operation, you can either:
  - set the participant's `OverwriteExistingFiles` setting to ***`True`***, or
  - delete, move, or rename the existing output files generated by CFD Server.

Perform remapping in the GUI

Right-click the **Mapping** branch and select **Map**.

Perform remapping in the CLI

1. Save the data model state to a temporary variable. (This example uses a variable named `savedState`.)

```
savedState = DatamodelRoot().GetState()
```

2. Clear the data model state. This clears the state inside System Coupling to allow re-mapping to take place.

```
ClearState()
```

3. Restore the data model state from the variable.

```
DatamodelRoot().SetState(State = savedState)
```

4. Reissue the mapping operation.

```
Solve()
```

Using Mapping Results

Once you are satisfied with the mapping results, you may use the `.csv` output file to provide displacements for a CFD analysis. To use the results in either a CFX or a Fluent analysis, you must import the `.csv` file as a profile.

- For details on using the results in a CFX analysis, see Establishing Profile Data and Creating Profile Functions in the *CFX-Pre User's Guide*.
- For details on using the results in a Fluent analysis, see Reading and Writing Profile Files in the *Fluent User's Guide*.

Electric Arc Modeling

System Coupling enables modeling of electric arc propagation by coupling Ansys Fluent and Ansys Electronics Desktop. System Coupling facilitates the exchange of heat losses, Lorentz forces, electrical conductivity, and temperature between these solvers in a coupled analysis workflow.

System Coupling provides a rich set of co-simulation features that are applicable to electric arc modeling analysis. For details, see the rest of this guide. This section provides only information that is specific to electric arc modeling.

For more information on coupled electric arc modeling analyses, see:

Coupling Participants

Set up your Fluent and Maxwell coupling participants and add them to the analysis.

Setting Up Coupling Participants

Set up your coupling participants and generate their coupling input files.

Fluent Setup and Input Requirements

When setting up Fluent:

- Make the Lorentz force and electrical conductivity variables (`lorentz-force` and `electrical-conductivity`, respectively) available by enabling an appropriate model (such as a battery model).
- Use a UDS to define electrical conductivity as a material property.
- Ensure that the variables are active on fluid volume regions.

When the setup is complete, export the `.scp` file and verify that it contains the above information.

Maxwell Setup and Input Requirements

When setting up Maxwell:

- Initialize data for the arc region and arc-related objects.
- For the arc region, ensure that electric conductivity is spatially dependent.
- For arc-related objects that have temperature-dependent materials, define initial temperatures.
- Assign an arc current.

When the setup is complete, generate the `.scp` file and verify that it contains the above information.

Adding Coupling Participants

Add the Fluent and Maxwell participants as described in Adding the Participants (p. 121).

Note that if the analysis is steady, it is an iteration-based analysis so long as all participants support this. For more information, see Steady Analysis Type (p. 37).

Coupling Interfaces

To exchange quantities required for electric arc modeling, you need to define a coupling interface between the corresponding arc regions within the CFD and electro-magnetic solver.

Adding a Coupling Interface

Add a coupling interface in the GUI

1. Right-click the **Setup** branch and select **Add Coupling Interface**.

   The coupling interface object is added to the data model under the **Coupling Interface** branch.

2. Verify that the two sides of the interface are defined correctly and adjust as needed.

Add a coupling interface in the CLI

Run the `AddInterface()`command, as in the following example:

```
AddInterface(
    SideOneParticipant = "FLUENT-1", SideOneRegions = ['fluid_arc_region'],
    SideTwoParticipant = "AEDT-2", SideTwoRegions = ['Region105'])
```

Adding Data Transfers

Data transfers define which variables are transferred from one side of the coupling interface to the other side. You may add data transfers using either System Coupling's GUI or CLI.

Add a data transfer in the GUI

In the GUI, right-click the coupling interface object and select **Add Data Transfer**.

Data transfers with relevant quantities are added.

Add a data transfer in the CLI

In the CLI, run the `AddDataTransfer()` command, as in the following example:

```
AddDataTransfer(
```

```
    Interface = interfaceName,
    TargetSide = 'Two',
    SourceVariable = " LorentzForce",
    TargetVariable = "lorentz-force")
```

System Coupling provides many options for general customizations of data transfers. For details, see the rest of the System Coupling documentation, especially:

- Expressions in System Coupling (p. 59) in this guide
- `DataTransfer (object)` in the *System Coupling Settings and Commands Reference*

Solving the Analysis

You may add solve the analysis using either System Coupling's GUI or CLI.

Solve the analysis in the GUI

In the toolbar, click the **Solve** button.

Solve the analysis in the CLI

Run the `Solve()` command, as in the following example:

```
Solve()
```

Monitoring and Postprocessing

For more information on monitoring the coupled solution and postprocessing the results, see:

- Using Parallel Processing Capabilities (p. 165)
- Monitoring Solution Progress (p. 133)
- Postprocessing Coupling Results in EnSight (p. 207)

Best Practices for System Coupling

When working with System Coupling, there are certain suggested practices that will help to ensure an efficient solution and to optimize the accuracy of your results. For best practices and setup recommendations, see:

General Best Practices for Coupled Analyses

This section provides best practices and recommendations that are generally applicable to all coupled analyses. For more information, see:

For recommendations on running coupled analyses for FSI applications, see Fluid-Structure Interaction (FSI) (p. 371).

Building Up to a Coupled Analysis

Once you have determined that a running a two-way coupled analysis is your objective, it is recommended that you build up to the final two-way analysis incrementally, progressively adding complexity with each step. The following sequence of steps is recommended for building any coupled analysis:

Step 1: Solve each participant problem individually as a decoupled analysis.

Before coupling your participant problems, verify that each one of them can be solved individually, as decoupled solutions. By running each one independently, you can get an approximation of their expected inputs to a coupled analysis.

As much as possible, try to set up the decoupled analyses so that they replicate the participants' expected roles in the prospective coupled analyses. (For example, to prepare for an FSI analysis, you would apply a pressure load in the structural analysis or set up mesh motion and deformation in the fluid analysis.)

Once you have verified that each of the problems can be solved individually on its own, you can proceed to working with one-way coupled analyses.

Step 2: Solve each participant problem as part of a one-way coupled analysis.

Verify that each of the participant problems can be run successfully as part of a one-way coupled analysis. To do this, create a one-way coupled analysis for each participant problem and use the results of the other participant's decoupled run as inputs, approximating the values it may receive in a two-way coupled analysis. This allows you to assess the sensitivity of each participant's analysis to the expected inputs.

The one-way coupled analyses should replicate the effects of the intended two-way coupled analysis as closely as possible. If one or both problems show sensitivity to the inputs it receives, then further exploration via a two-way coupled analysis may be useful.

Once the one-way analyses have been solved, how you proceed depends on your end goal for the problem:

- If a one-way coupled analysis is your end goal, then the next step is to evaluate its convergence. For information on assessing coupled analysis convergence, see Evaluating Convergence and Data Transfer Accuracy (p. 397). For participant-specific methods of evaluating convergence, see the relevant participant product documentation.
- If a two-way coupled analysis is your end goal, then proceed to the creation of the two-way analysis.

Step 3: Solve both participant problems as a two-way coupled analysis.

Solve both participant problems as a two-way coupled analysis. To do this, remove the approximated inputs used for the one-way analyses and then incorporate each participant problem into the two-way simulation.

For an illustration of how these steps might be used for a specific application, see Building a Coupled FSI Analysis from Decoupled Participant Problems (p. 371).

Improving Mapping Quality

When a coupled analysis is executed, System Coupling executes a mapping process, using data from source-mesh locations to calculate data for target-mesh locations. When mapping is performed, the process runs to completion and generates data regardless of any existing issues that may impact the quality of the mapping. Use the following best practices to help ensure the accuracy of the mapping and the resulting data transfer:

Review mapping diagnostics.

Mapping diagnostics written to the Mapping Summary (p. 342) of the Transcript/Log file provide insight into the ultimate accuracy of the data transfers. They show the percentage of the target and source mesh locations that were successfully mapped, providing insight into the ultimate accuracy of the data transfers.

- **Target-side diagnostics:**
  - **If less than 100%** of the target-side locations were mapped, then System Coupling is implementing a method of generating data on target locations according to the type of mapping used for the transfer.

This is often caused by differences in the geometry. For example, mapping might be prevented because the target geometry does not overlap/intersect the source geometry, or there might be a gap (too much distance) between the source and target geometries.

For more information, see Generating Data for Target Locations (p. 289).

– **If between 99% and 100%** of the target-side locations were mapped, then this is shown in the diagnostics as **>99%** and all other values are rounded to the nearest whole number.

- **Source-side diagnostics:**

  – **If less than 100%** of the source-side locations were mapped, then some source data is not being used.

  In conservative mapping, this means that the quantity transferred is either not fully conserved and/or is conserved only in areas where the mesh overlaps. However, source-side mapping of less than 100% typically occurs only in cases of profile-preserving mapping where the source mesh is significantly finer than the target mesh.

  – **If between 99% and 100%** of the source-side locations were mapped, then this is shown in the diagnostics as **>99%** and all other values are rounded to the nearest whole number.

Use similar participant geometries.

Use geometries that are as similar as possible on all participant regions on the coupling interface. For example:

- Minimize the gap between the source and target geometries, if possible.
- Ensure that there is overlap/intersection between the source and target geometry regions where data will be transferred.

  – For data transfers using **profile-preserving mapping to a target surface**, you may address areas of non-overlap by changing the value of the `UnmappedValueOption` setting to ***`Extrapolation`***. With this setting, target data is smoothly extrapolated onto the non-overlapping sections of the target surface.

- Ensure that all source and target regions on the coupling interface have the same reference state.

Take advantage of mapping controls.

- By default, System Coupling checks the quality of intersection between the sides of a coupling interface. An interface-level `StopIfPoorIntersection` setting allows you to specify that System Coupling stops and generates a setup error when intersection quality falls below a specified level, which you can change using the `PoorIntersectionThreshold` setting.
- If gaps cannot be eliminated for surface transfers (for example, a structural geometry is in its manufactured state and a fluid geometry is in its deformed state), mapping accuracy may be improved by adjusting the `AbsoluteGapTolerance` and/or the `RelativeGapTolerance`.
- For surface transfers, ensure that the surface mesh for each participant has area vectors which point out of the volume region associated with that participant. You can visualize these area

vectors in EnSight using the **area__EV** variable. In the following rare situations, the area vector might point into the volume region rather than out:

– The surface region comes from a Fluent fluid-porous interface.

– The surface region comes from the solid side of a conjugate heat transfer (CHT) interface in Fluent or CFX.

– The mesh does not satisfy usual Ansys mesh ordering conventions. This should not happen with meshes generated by Ansys meshing products but may be possible if the mesh was generated using an external tool.

If any of these situations occurs, you may improve the mapping by setting the `FaceAlignment` setting to:

– ***`OppositeOrientation`*** (same as default ***`ProgramControlled`*** option except when MAPDL shells are present)

– ***`AnyOrientation`*** (may fail if the geometry has thin bodies)

– ***`SameOrientation`*** (if you know the area vectors are directed inward)

- For surface transfers to one side of an MAPDL shell region, if the geometry also has thin bodies (for example, thin airfoils constructed from shells), you may need to inspect the area vectors (**area__EV**) in EnSight to determine which way they are oriented, and then set `FaceAlignment` to either ***`SameOrientation`*** (if vectors are oriented into the shell) or ***`OppositeOrientation`*** (if vectors are oriented out of the shell).

- For surface transfers to both sides of an MAPDL shell region, System Coupling will give accurate mapping results only for conservative transfers to the shell region and profile-preserving transfers from the shell region.

Use similar participant meshes.

Use meshes that are as similar as possible on all participant regions on the coupling interface. For example:

- Refine meshes to the degree feasible. Coarse meshes can cause and/or exacerbate mesh gaps or dissimilarities, which in turn can have an adverse effect on mapping. This is especially important for geometries with meshes on curved surfaces.

- Ensure that the meshes of the source and target regions of the coupling interface are suitable for the problem being solved. In general, source and target mesh resolutions should be as similar as is possible. Variances should be guided primarily by the participant physics and the type of data being transferred. For example:

  – For **data transfers of intensive variables** (that is, which use a profile-preserving mapping algorithm), a target-mesh resolution that is *similar to* or *finer than* the source-mesh resolution is recommended.

    When a profile-preserving algorithm is used, the transferred quantity is better preserved when the target mesh has either the same number or more nodes than the source mesh. This prevents features that are resolved on the source mesh from being lost in the transfer to the coarser target mesh.

– For **data transfers of extensive variables** (that is, which use a conservative mapping algorithm), a target mesh that is *similar to* the source mesh is recommended, though a target mesh that is slightly *coarser than* the source mesh is acceptable.

  When a conservative algorithm is used, there is no benefit (from a mapping point of view) in the target mesh having more nodes than the source mesh. The conservative nature of the transfer ensures that the quantity is preserved even when transferred to a coarser target mesh.

  Note, however, that a finer target mesh may be beneficial from a physics-modeling point of view. Mesh resolution should be adjusted as needed to obtain an accurate solution for the participant physics.

Use geometries that are similarly oriented.

Ensure that participant geometries have similar orientations. Significant differences may cause the intersection between coupling interface sides to be insufficient for accurate mapping between them. System Coupling provides the following ways to minimize orientation-based issues:

- Determine when orientation issues affect mapping.
- Improve the intersection between participant geometries.

  Use System Coupling's geometry transformation functionality to define transformation reference frames.

Note:

In most cases, reference-state disparities must be addressed as part of the geometry and/or participant setup. For thermal-electromagnetic problems, however, they can be addressed by System Coupling during the coupled analysis setup.

Evaluating Convergence and Data Transfer Accuracy

Use the following best practices to evaluate the convergence and the accuracy of data transfers:

Set an appropriate convergence target.

Set data transfer and participant convergence target that are strict enough to ensure that the data on the target is not affected by the input data once the convergence criteria have been reached.

Review convergence diagnostics.

Review convergence diagnostics for each coupling iteration (p. 346) of each coupling step. The coupling step/iteration sections of the Transcript/Log show the convergence status for data transfers and participants. Data transfers, and, in most cases, participants should reach convergence in each coupling step.

- **Data transfer diagnostics:**

Review the data transfer diagnostics reported at analysis initialization (p. 341) and for each coupling iteration (p. 346) of each coupling step. The coupling step/iteration sections of the Transcript/Log show the values data transfer values reported by the source and target participants. The **Sum** is reported for extensive variables (mapped using a conservative algorithm) and the **Weighted Average** (over all regions in the interface) is reported for intensive variables (mapped using a profile-preserving algorithm).

The reported values should be the same for both the source and the target of the data transfer. When they are not:

– If the **target sum is less than the source sum**, then this indicates (and should confirm) that System Coupling is implementing a method of generating data on unmapped target locations according to the type of mapping used for the transfer.

For more information, see Generating Data for Target Locations (p. 289).

– If the ***target weighted average significantly differs from the source weighted average***, then the mapped results are suspect and should be reviewed in greater detail. This can often be attributed to the target-side meshes that are coarse relative to the source-side meshes, especially in regions with significant source-solution variations.

- **Participant diagnostics:**

The participant-solution status section indicates whether each participant completed its step during the iteration.

Improving the Accuracy of Steady Analyses

Use the following best practices to improve the accuracy of steady coupled analyses.

Define an appropriate number of coupling steps/iterations.

Best practices for choosing the number of coupling steps and/or iterations vary according to whether the analysis is iteration-based or step-based. (For more information, see Steady Analysis Type (p. 37).)

- **Iteration-based steady analyses:**

An iteration-based steady analysis allows you to define the minimum and maximum number of iterations. Restart points can be created at the end of each coupling iteration. Ramping and under-relaxation can be applied across iterations.

Ensure that you define enough coupling iterations to allow the analysis to converge. For more information, see Assess the convergence status of the analysis (p. 399).

- **Step-based steady analyses:**

A step-based steady analysis can have one or more coupling steps, each with multiple steps permitted. Restart points can be created only at the end of each coupling step.

Set the number of coupling steps and coupling iterations according to how you need to balance requirements for restarting the analysis, minimizing file-storage space, and/or using System Coupling's ramping (p. 300) and under-relaxation (p. 302) algorithms.

Ramping capabilities and setup recommendations vary according to the number of coupling steps, as follows:

– **One coupling step:**

→ Because restart data are generated only at the end of a completed coupling step, a single-step analysis can only be restarted at the end of the solution. As such, it is not possible to restart from an intermediate result or an abnormally terminated run.

→ Minimizes storage space at the expense of restart capabilities.

→ Allows ramping and under-relaxation across coupling iterations within the single coupling step.

– **Multiple coupling steps:**

→ Restart data are still generated only at the end of coupling steps, but you can specify which steps for which restart data will be generated. As such, it is possible to restart from an intermediate result or from an abnormally terminated run, which can be useful for complex analyses that are at risk of abnormal terminations.

→ Allows multiple restarts at the expense of storage space.

→ Allows ramping and under-relaxation across coupling iterations within a coupling step, but not across coupling steps. When a step is completed, the full data transfer value is transferred to the next step.

Assess the convergence status of the analysis.

Identify and address any issues that may be affecting the convergence of the analysis.

- **Slow convergence:**

  If convergence is not realized within tens of coupling iterations (that is, up to 100), then consider introducing stabilization methods such as ramping (p. 300) and under-relaxation (p. 302).

  Participant-specific solution stabilization methods, such as those offered by Fluent, may also be employed. For details, refer to the corresponding participant product documentation.

- **Strong non-linearities:**

  If there are strong non-linearities within the coupled solution, then more coupling iterations may be required, even with the use of stabilization. You can increase the maximum number of coupling iterations per step. For step-based steady analyses, you can also try introducing additional coupling steps.

- **Premature convergence:**

  In some cases, convergence is realized prematurely (for example, within one iteration) because the non-linearities in the coupled solution have not yet been activated. If this is suspected, then set a minimum number of coupling iterations per step to trigger the non-linearities and force adequate resolution.

Improving the Accuracy of Transient Analyses

Use the following best practices to improve the accuracy of transient coupled analyses (p. 38).

Define an appropriate coupling step size.

Set the coupling step size to resolve the coupling-related timescales. For example, with thermal couplings, choose a coupling time step size that is long enough to resolve the thermal timescales.

- If a participant's timescales are **slightly smaller** (for example, between 1x and 100x) than coupling-related timescales, then reduce the coupling step size to resolve to the smallest timescale in the participant physics. This often occurs for fluid-structure interaction applications.
- If a participant's timescales are **significantly smaller** (for example, greater than 100x) than coupling-related timescales, then run that participant using a steady analysis. This often occurs for electromagnetic/thermal applications.

Define motion consistently across coupling participants.

When motion is defined on a coupled region ensure that the motion is defined consistently for both coupling participants. (Currently, this is supported only for electromagnetic-thermal analyses between Maxwell and Fluent coupling participants.)

Assess the convergence status of the analysis.

Identify and address any issues that may be affecting the convergence of the analysis.

- **Slow convergence:**

  If convergence is not realized for each coupling step within approximately 5 – 10 coupling iterations, then consider reducing the coupling time step size or introducing stabilization methods such as ramping (p. 300) and under-relaxation (p. 302).

  Participant-specific solution stabilization methods, such as those offered by Fluent, may also be employed. For details, refer to the corresponding participant product documentation.

- **Strong non-linearities:**

  If there are strong non-linearities within the coupled solution, then convergence within approximately 5 – 10 coupling iterations may not be possible. If this is suspected (and the previous recommendations are not sufficient to allow convergence), then increase maximum number of coupling iterations per step.

  Note:

  Although maximum number of iterations may be used to limit the amount of work done per step, this is generally not recommended.

- **Premature convergence:**

  In some cases, convergence is realized prematurely (for example, within one iteration) because the non-linearities in the coupled solution have not yet been activated. If this is suspected,

then set a minimum number of coupling iterations per step to trigger the non-linearities and force adequate resolution.

Improving Coupled Analysis Stability

If the physics in a coupled analysis are strongly coupled, convergence difficulties may result. There are a number of tools which may help stability. Consider using these tools in the following order of priority:

Quasi-Newton Solution Stabilization

Data Transfer Ramping

Data Transfer Relaxation

Participant Solution Stabilization

Quasi-Newton Solution Stabilization

For certain two-way thermal and FSI coupled analyses, convergence instabilities can be addressed with System Coupling's ***Quasi-Newton solution stabilization and acceleration method***.

Quasi-Newton stabilization, which is based on the *Interface Quasi-Newton – Inverse Least Squares (IQN-ILS)*[1] (p. 304) algorithm, can be used to address convergence issues in coupled analyses run in one of System Coupling's user interfaces. It can improve convergence by applying an approximate Newton iteration to the coupling interface data.

For more detailed information, see Quasi-Newton Stabilization Algorithm (p. 303).

Data Transfer Ramping

If the full magnitude of the data transfer is applied to the target side on the first iteration, it may be too much of a change for the next coupling participant to absorb instantaneously. To soften the startup, you can ramp the target-side data from the final value in the previous coupling step to the full magnitude during the initial coupling iterations within the current step. For more information about the ramping algorithm, see Ramping Algorithm (p. 300).

For more information about ramping controls in a given context, see:

- the `RampingOption` data model setting (for System Coupling in its GUI or CLI), as described in the *System Coupling Settings and Commands Reference* manual.
- the **Ramping** option in the Data Transfer Control (p. 252) settings (for System Coupling in Workbench) or

---

Note:

On the first coupling iteration of the first step (including after a restarted run), the ramping algorithm ramps against the initial guess for the data transfer. To obtain this initial guess, System Coupling interpolates the initial guess of the participant which serves this initial data. An alpha-level setting allows you to revert to System Coupling's previous method of setting initial target-side reference values to program-specified constants. To do so, perform the following steps:

1. Enable alpha-level settings, as described in Activate System Coupling Hidden Features in the *System Coupling Beta Features* documentation.
2. Set `AnalysisControl.TargetInitializationOption` to ***`UseConstantValue`***.

For details about the constant values that will be applied, see Initial Values Used for the Reference Target-Side Value in the 2020 R1 *System Coupling User's Guide*.

Data Transfer Relaxation

Another way to slow down the rate of change in data transfers is to explicitly under-relax the transfers, as described in Under-Relaxation Algorithm (p. 302).

Data transfer relaxation differs from ramping in two details. First, it is applied every iteration, rather than over the minimum iterations, so the final source-side value is applied to the target side only upon convergence. Second, relaxation is applied against the previous iteration value rather than the value from the end of the previous coupling step.

Data transfer relaxation differs from the Quasi-Newton stabilization algorithm in that the relaxation factor must be set by the user. You can think of the Quasi-Newton algorithm as a sophisticated type of dynamic relaxation which varies per-iteration and per-transfer location.

Participant Solution Stabilization

Solution instabilities that manifest as a very rapid divergence of the coupled analysis may arise if a given coupling participant is particularly sensitive to data obtained from another participant. In these cases, it may be helpful to use various solution stabilization algorithms that have been implemented in the target participant.

For an example of participant solution stabilization, refer to the dynamic mesh System Coupling solution options used in Fluent, described in System Coupling Motion in the Fluent User's Guide.