

ANSYS ACT Customization Guide for Workbench



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 2020 R2
July 2020

ANSYS, Inc. and
ANSYS Europe,
Ltd. are UL
registered ISO
9001:2015
companies.

Copyright and Trademark Information

© 2020 ANSYS, Inc. Unauthorized use, distribution or duplication is prohibited.

ANSYS, ANSYS Workbench, AUTODYN, CFX, FLUENT and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries located in the United States or other countries. ICEM CFD is a trademark used by ANSYS, Inc. under license. CFX is a trademark of Sony Corporation in Japan. All other brand, product, service and feature names or trademarks are the property of their respective owners. FLEXlm and FLEXnet are trademarks of Flexera Software LLC.

Disclaimer Notice

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with exporting laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement.

ANSYS, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001: 2015 companies.

U.S. Government Rights

For U.S. Government users, except as specifically granted by the ANSYS, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the ANSYS, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

Third-Party Software

See the [legal information](#) in the product help files for the complete Legal Notice for ANSYS proprietary software and third-party software. If you are unable to access the Legal Notice, contact ANSYS, Inc.

Published in the U.S.A.

Table of Contents

Introduction	1
Workbench APIs	3
Accessing Project-Level APIs	3
Accessing Task APIs	4
Accessing Task Template APIs	6
Accessing Task Group APIs	10
Accessing Task Group Template APIs	11
Accessing Parameter APIs	12
Accessing Property APIs	13
Accessing State-Handling APIs	14
Accessing Project Reporting APIs	16
Workbench Feature Creation	23
Custom User-Specified Interface Operation	23
Creating the Extension for Custom Interface Operations	24
Defining Functions for Custom Interface Operations	24
Global Workflow Callbacks	25
Creating the Extension for Global Callbacks	25
Defining Functions for Global Callbacks	26
ACT-Based Property Parameterization in Workbench	28
Simulation Workflow Integration	31
Custom Task Group and Task Exposure in the Project Schematic	31
ACT Workflow Designer	32
Defining Properties for Workflow Cells	36
Publishing and Resetting Workflows	37
Publishing the Workflow	37
Resetting the Workflow	39
Defining Task Input and Output Properties	40
Creating an Example Custom Workflow for a “Squares” Application	43
Global Callbacks	46
Invoking Custom Project Schematic Actions	47
Invoking Custom Workbench Actions	47
Manipulating the Observation Process of Global Callbacks	48
Data Transfers	48
Progress Monitoring	49
Process Utilities	51
Manual Creation of a Custom Workflow	52
XML Extension Definition for a Custom Workflow	53
Defining a Global Callback	56
Invoking Global Callbacks	57
Filtering Global Callbacks	57
Specifying Global Callback Execution Order	58
Available Project Schematic Callbacks	58
Available Workbench Callbacks	61
Defining a Task	61
Defining Task-Level Attributes	63
Defining Task-Level Data Transfer	63
Defining Task-Level Data Transfer Access	65
Managing Task-Level Files	67
Defining Task-Level Callbacks	68

Defining Task-Level Context Menus	69
Defining Task-Level Property Groups and Properties	69
Defining Task-Level Property Callbacks	70
Using Standardized Property Interaction	74
Using Path-Based Property Control Support	75
Using String-Based and Option-Based Task Properties	75
Specifying Property Default Values	76
Defining Task-Level Parameters	76
Defining Task-Level Inputs and Outputs	76
Defining a Remote Job Execution Specification	78
Accessing and Invoking Task-Related Workbench Data	79
Executing Container-Level Commands on a Task	79
Accessing Workbench Parameters from a Task	79
Accessing an ANSYS-Installed Component from a Task	79
Accessing the Owing Task Group from a Task	80
Defining a Task Group	81
IronPython Script for a Custom Workflow	83
Upstream Data Consumption (Input)	83
Data Generation (Output)	83
Convenience APIs	84
Simulation Workflow Integration Examples	87
Parametric Task Group	87
Creating the Extension for Creating a Parametric Task Group	87
Defining Functions for Creating a Parametric Task Group	88
External Application Integration with Parameter Definition	88
Creating the Extension for Integrating an External Application with Parameter Definition	89
Defining Functions for Integrating an External Application with Parameter Definition	90
External Application Integration with Custom Data and Remote Job Execution	92
Creating the Extension for Integrating an External Application with Custom Data	92
Defining Functions for Integrating an External Application with Custom Data	95
Using RSM Job Submission Capabilities	97
Generic Mesh Transfer	99
Creating the Extension for Generic Mesh Transfer	99
Defining Functions for Generic Mesh Transfer	101
Custom Transfer	101
Creating the Extension for Custom Transfer	102
Defining Functions for Custom Transfer	103
Generic Material Transfer	103
Creating the Extension for Generic Material Transfer	104
Defining Functions for Generic Material Transfer	105
Reviewing the Referenced Material File	106
Workbench Wizards	109
A. Component Input and Output Tables	111
B. ANSYS-Installed System Component Template and Display Names	187
C. Data Transfer Types	209
D. Addin Data Types and Data Transfer Formats	215
E. ANSYS-Installed Custom Workflow Support	219

List of Tables

1. 3D ROM	111
2. ACP (Post)	111
3. ACP (Pre)	112
4. Autodyn	113
5. BladeGen	113
6. CFX	113
7. CFX (Beta)	114
8. Chemkin (Beta)	114
9. Coupled Field Static	114
10. Coupled Field Transient	116
11. Data Receive	117
12. Data Receive (Beta)	117
13. Data Send	117
14. Designer Circuit	118
15. Designer Circuit Netlist	118
16. Direct Optimization	118
17. Eigenvalue Buckling	118
18. EigenValue Buckling (Samcef)	120
19. Electric	121
20. EnSight (Forte)	122
21. Engineering Data	122
22. Explicit Dynamics	122
23. External Connection	124
24. External Data	124
25. External Model	124
26. Feedback Iterator	124
27. Fluent	124
28. Fluent (with CFD-Post) (Beta)	125
29. Fluent (with Fluent Meshing)	125
30. Fluid Flow (CFX)	126
31. Fluid Flow (Fluent)	127
32. Fluid Flow (Polyflow)	128
33. Fluid Flow – Blow Molding (Polyflow)	129
34. Fluid Flow – Extrusion (Polyflow)	130
35. Forte	130
36. GRANTA MI	131
37. Geometry	131
38. HFSS	131
39. HFSS 3D Layout Design	131
40. HFSS-IE	132
41. Harmonic Acoustics	132
42. Harmonic Response	133
43. Hydrodynamic Diffraction	134
44. Hydrodynamic Response	135
45. IC Engine (Fluent)	135
46. ICEM CFD	136
47. Icepak	137
48. Injection Molding Data (Beta)	137
49. MOP Solver	137

50. Magnetostatic	137
51. Material Designer	138
52. Maxwell 2D	139
53. Maxwell 3D	139
54. Mechanical APDL	139
55. Mechanical Model	140
56. Mesh	141
57. Microsoft Office Excel	141
58. Modal	141
59. Modal (ABAQUS)	142
60. Modal (NASTRAN) (Beta)	144
61. Modal (Samcef)	145
62. Modal Acoustics	146
63. Optimization	147
64. Parameters Correlation	147
65. Performance Map	148
66. Polyflow	148
67. Polyflow – Blow Molding	148
68. Polyflow – Extrusion	148
69. Python (Beta)	149
70. Q3D 2D Extractor	149
71. Q3D Extractor	149
72. RMXprt	150
73. Random Vibration	150
74. Response Spectrum	151
75. Response Surface	152
76. Response Surface Optimization	152
77. Results	153
78. Rigid Dynamics	153
79. Robustness	154
80. SPEOS	154
81. Sensitivity	155
82. Sherlock (Post) (Beta)	155
83. Sherlock (Pre) (Beta)	155
84. Signal Processing	155
85. Six Sigma Analysis	155
86. Static Acoustics	156
87. Static Mechanical	157
88. Static Structural	159
89. Static Structural (ABAQUS)	160
90. Static Structural (Samcef)	161
91. Steady-State Thermal	163
92. Steady-State Thermal (ABAQUS)	164
93. Steady-State Thermal (MUSCADE)	165
94. Steady-State Thermal (Samcef)	166
95. System Coupling	168
96. Thermal-Electric	168
97. Throughflow	169
98. Throughflow (BladeGen)	170
99. Topology Optimization	170
100. Transient Mechanical	172

101. Transient Structural	173
102. Transient Structural (ABAQUS)	174
103. Transient Structural (Samcef)	176
104. Transient Thermal	177
105. Transient Thermal (ABAQUS)	178
106. Transient Thermal (Samcef)	180
107. Turbo Setup	181
108. TurboGrid	181
109. Turbomachinery Fluid Flow	181
110. Turbomachinery Fluid Flow (BladeEditor) (Beta)	182
111. Turbomachinery Fluid Flow (BladeGen) (Beta)	183
112. Twin Builder	183
113. Vista AFD	183
114. Vista CCD	184
115. Vista CCD (with CCM)	184
116. Vista CPD	184
117. Vista RTD	184
118. Vista RTD (Beta)	184
119. Vista TF	184
120. OptiSLang Pre (Beta)	185
121. 3D ROM	187
122. ACP (Post)	187
123. ACP (Pre)	187
124. Autodyn	187
125. BladeGen	187
126. CFX	187
127. CFX (Beta)	188
128. Chemkin (Beta)	188
129. Coupled Field Static	188
130. Coupled Field Transient	188
131. Data Receive	189
132. Data Receive (Beta)	189
133. Data Send	189
134. Designer Circuit	189
135. Designer Circuit Netlist	189
136. Direct Optimization	189
137. Eigenvalue Buckling	189
138. Eigenvalue Buckling (Samcef)	190
139. Electric	190
140. EnSight (Forte)	190
141. Engineering Data	190
142. Explicit Dynamics	190
143. External Connection	191
144. External Data	191
145. External Model	191
146. Feedback Iterator	191
147. Fluent	191
148. Fluent (with CFD-Post) (Beta)	191
149. Fluent (with Fluent Meshing)	191
150. Fluid Flow (CFX)	191
151. Fluid Flow (Fluent)	192

152. Fluid Flow (Polyflow)	192
153. Fluid Flow - Blow Molding (Polyflow)	192
154. Fluid Flow - Extrusion (Polyflow)	192
155. Forte	192
156. GRANTA MI	193
157. Geometry	193
158. HFSS	193
159. HFSS 3D Layout Design	193
160. HFSS-IE	193
161. Harmonic Acoustics	193
162. Harmonic Response	194
163. Hydrodynamic Diffraction	194
164. Hydrodynamic Response	194
165. IC Engine (Fluent)	194
166. ICEM CFD	195
167. Icepak	195
168. Injection Molding Data (Beta)	195
169. MOP Solver	195
170. Magnetostatic	195
171. Material Designer	195
172. Maxwell 2D	195
173. Maxwell 3D	196
174. Mechanical APDL	196
175. Mechanical Model	196
176. Mesh	196
177. Microsoft Office Excel	196
178. Modal	196
179. Modal (ABAQUS)	196
180. Modal (NASTRAN) (Beta)	197
181. Modal (Samcef)	197
182. Modal Acoustics	197
183. Optimization	198
184. Parameters Correlation	198
185. Performance Map	198
186. Polyflow	198
187. Polyflow - Blow Molding	198
188. Polyflow - Extrusion	198
189. Python (Beta)	198
190. Q3D 2D Extractor	198
191. Q3D Extractor	199
192. RMxport	199
193. Random Vibration	199
194. Response Spectrum	199
195. Response Surface	199
196. Response Surface Optimization	199
197. Results	200
198. Rigid Dynamics	200
199. Robustness	200
200. SPEOS	200
201. Sensitivity	200
202. Sherlock (Post) (Beta)	200

203. Sherlock (Pre) (Beta)	200
204. Signal Processing	201
205. Six Sigma Analysis	201
206. Static Acoustics	201
207. Static Mechanical	201
208. Static Structural	201
209. Static Structural (ABAQUS)	202
210. Static Structural (Samcef)	202
211. Steady-State Thermal	202
212. Steady-State Thermal (ABAQUS)	202
213. Steady-State Thermal (MUSCADE)	203
214. Steady-State Thermal (Samcef)	203
215. System Coupling	203
216. Thermal-Electric	203
217. Throughflow	203
218. Throughflow (BladeGen)	204
219. Topology Optimization	204
220. Transient Mechanical	204
221. Transient Structural	204
222. Transient Structural (ABAQUS)	204
223. Transient Structural (Samcef)	205
224. Transient Thermal	205
225. Transient Thermal (ABAQUS)	205
226. Transient Thermal (Samcef)	205
227. Turbo Setup	206
228. TurboGrid	206
229. Turbomachinery Fluid Flow	206
230. Turbomachinery Fluid Flow (BladeEditor) (Beta)	206
231. Turbomachinery Fluid Flow (BladeGen) (Beta)	206
232. Twin Builder	206
233. Vista AFD	207
234. Vista CCD	207
235. Vista CCD (with CCM)	207
236. Vista CPD	207
237. Vista RTD	207
238. Vista RTD (Beta)	207
239. Vista TF	207
240. OptiSLang Pre (Beta)	207
241. Data Transfer Types and Properties	209
242. Task Groups and Tasks	219

Introduction

This guide assumes that you are familiar with the general ACT usage information in the [ACT Developer's Guide](#). It supplies only information specific to using ACT to customize Workbench.

This guide organizes content into primary sections as follows:

- [Workbench APIs \(p. 3\)](#): Describes the ACT APIs that provide for integrating external applications and custom processes into the Workbench workflow.
- [Workbench Feature Creation \(p. 23\)](#): Describes how to define ACT-based task properties as input parameters, output parameters, or non-parameterized values.
- [Simulation Workflow Integration \(p. 31\)](#): Describes custom ACT workflows and the **ACT Workflow Designer**, which automates workflow setup in Workbench. Also provides technical information and the manual creation process.
- [Simulation Workflow Integration Examples \(p. 87\)](#): Describes supplied extensions that integrate custom workflows in Workbench.
- [Workbench Wizards \(p. 109\)](#): Describes a Workbench project wizard, using a supplied extension as an example.

Note:

For information on all ACT API changes and known issues and limitations that may affect your existing ACT extensions, see [Migration Notes](#) and [Known Issues and Limitations](#) in the *ANSYS ACT Developer's Guide*.

Workbench APIs

ACT provides APIs for creating custom workflows in the Workbench **Project Schematic**. Custom workflow APIs provide access to items in the **Project Schematic**, including the framework data model, interface components, and user objects and their properties.

At the workflow level, you can use global callbacks to invoke **Project Schematic** actions (both ANSYS-installed and ACT-defined). You can also create and expose custom *task groups* (systems) and *tasks* (components) that can interact with each other and with ANSYS-installed components and systems as part of the **Project Schematic** workflow.

The following topics provide access information:

[Accessing Project-Level APIs](#)

[Accessing Task APIs](#)

[Accessing Task Template APIs](#)

[Accessing Task Group APIs](#)

[Accessing Task Group Template APIs](#)

[Accessing Parameter APIs](#)

[Accessing Property APIs](#)

[Accessing State-Handling APIs](#)

[Accessing Project Reporting APIs](#)

Accessing Project-Level APIs

You can access project-level APIs off the master **ExtAPI** entry point.

Project

The top level of the hierarchy is the framework-level project. It provides additional entry points to the Workbench data model, file manager, and so on. It is accessed as follows:

```
project = ExtAPI.DataModel.Project
```

Context

This is the framework-level command context. It provides additional entry points to the session and project utilities. It is accessed as follows:

```
context = ExtAPI.DataModel.Context
```

Tasks

This is all of the tasks in the **Project Schematic**, which includes both ACT-defined and ANSYS-installed tasks. It does not include the “master task” level, such as ACT and **Project Schematic** “template-like” tasks. Only concrete task instances are included. It is accessed as follows:

```
tasks = ExtAPI.DataModel.Tasks
```

Task Groups

This is all of the task groups in the **Project Schematic**, which includes both ACT-defined task groups and ANSYS-installed task groups. It does not include the “master task group” level, such as ACT and **Project Schematic** “template-like” task groups. Only concrete task group instances are included. It is accessed as follows:

```
taskGroups = ExtAPI.DataModel.TaskGroups
```

Note:

This section provides only the top-level API access points to **Project Schematic** and workflow-related data. For a comprehensive list of APIs for custom workflows, see the [<workflow>](#) section of the [ANSYS ACT API Reference Guide](#).

Accessing Task APIs

Workbench tasks are exposed through Automation API wrappers in the namespace **Ansys.ACT.Workbench.Automation.Workflows**. The project data model returns this wrapper type:

```
task = ExtAPI.DataModel.Tasks[0]
```

The task wrapper does not store any local data. All data access and modification occur at member call time.

For the class **Task**, property access is provided through a dictionary. You access a property wrapper by name to get or set values. For example, the Geometry component exposes the property **ImportSolidBodies**. You can access and change this property as follows:

```
task = ExtAPI.DataModel.Tasks[0]
task.Properties["GeometryImportSolidBodies"].Value = True
```

Note:

ACT-defined callbacks still receive **UserTasks** as arguments. A **UserTaskTemplate** is now accessible from the property **Template**. All other task functionality remains intact.

The following table lists task APIs.

Class	Member	Description
ITask	Name	The internal task name.
	Caption	The user-visible task display text.
	TaskGroup	The parent task group.
	InternalObject	The task's internal object.
Task	TaskId	The internal Workbench component identifier.
	Caption	The Workbench UI-visible display text.

Class	Member	Description
	TaskGroup	The task group wrapper for the component's system instance.
	InternalObject	The Workbench component.
	SourceTasks	The list of tasks providing data to this task.
	TargetTasks	The list of tasks consuming data from this task.
	State	The task state.
	Template	Gets the template from which the task was created.
	UserId	Gets the ID to display in the UI.
	DirectoryName	Gets the data directory name to display in the UI.
	MasterTask	Gets the master task from which the task directly consumes shared data. This is always the immediate upstream sharing source, even if the sharing is full-share. For the true root data task, use RealTask .
	RealTask	Gets the shared data root task from which the task consumes shared data. This is always the first (original, non-grayed) sharing source if fully shared. Otherwise, it is the task itself.
	IsSharing	Gets a value indicating whether the task is a slave of a master-sharing source.
	Container	Gets the container for the task.
	Notes	Gets the task notes.
	UsedLicenses	Gets the used license for the component. This dictionary maps the license IDs to the number of used licenses for the task.
	Solver	Gets the task solver type.
	ImageName	Gets the name for the task icon image.
	IsFailedState	Gets a value indicating whether the task is in a failed state from a previous data operation, such as update or refresh.
	AlwaysBePartOfDpUpdate	Gets a value indicating whether the task should always be part of a design point update, even if it does not affect any parameter value.
	CoupledClients	Gets the list of coupled client data tasks.
	CoupledToTask	Gets the coupling server task (if available).
	CoordinateId	Gets the task coordinate ID as a union of the task group letter and task index.
	Parameters	The list of input and output parameters associated with the task.

Class	Member	Description
	Properties	Gets a dictionary of properties, accessible by property name.
	InputData	Gets the input data, by type string key, for the task.
	OutputData	Gets the output data, by type string key, for the task.
	GetPropertyNames()	Gets the names of all properties declared with the task's data model.
	ExecuteCommand(command Name, arguments)	Executes a specific task-bound command.
	Clean()	Deletes the heavyweight data of this task to reduce the size of the project. Heavyweight data can include the solution, results, or both.
	Refresh()	Refreshes the input data for a task by reading all changed data from upstream (source) tasks. No calculations or updates based on this new data are performed.
	Reset()	Resets the task by removing all user input and result data.
	Update()	Updates the task by refreshing the input from all upstream components and then performs a local calculation based on current data.
	Remove()	Deletes the task and all dependents from its task group.

Accessing Task Template APIs

Workbench task templates are exposed through Automation API wrappers in the namespace **Ansys.ACT.WorkBench.Automation.Workflows**. The project data model returns this wrapper type:

```
task = ExtAPI.DataModel.Tasks[0]
template = task.Template
```

The task template wrapper does not store any local data. All data access and modification occur at member call time.

The following table lists task template APIs.

Class	Member	Description
TaskTemplate	Name	Gets the task template name.
	TaskTypeAbbreviation	Gets the default name for the task.
	DisplayName	Gets the user-visible name of the task created from the template. If the task group

Class	Member	Description
		template gives a name for the task using the property TaskNames , it takes precedence.
	Solver	Gets the solver type in the task. If specified, this contributes to property Solver for the containing task group.
	IsResults	Gets a value indicating whether the task represents result data, which is then not duplicated when the DuplicateSystem-WithoutResults is invoked.
	IsShareable	Gets a value indicating whether one instance of this type of task can be shared by another task.
	IsReplaceable	Gets a value indicating whether sharing of this type of task can be changed dynamically from sharing to unsharing or vice versa.
	IsReplaceableWith-Sharing	Gets a value indicating whether sharing of this type of task can be changed dynamically from an independent task to a full-sharing task. This covers one of the two aspects of what IsReplaceable defines.
	IsDeletable	Gets a value indicating whether this type of task and its downstream tasks can be deleted from its task group.
	IsPartialShareOnly	Gets a value indicating whether this type of task engages in only partial sharing.
	DontShareByDefault	Gets a value indicating whether this type of task has all connections established as "provides" connections to new systems (instead of sharing).
	RequiredInputTypes	Gets a list of types that are required as inputs for the task to be valid.
	OutputTypes	Gets a list of types that are contained in the task after creation.
	EquivalentInputs	Gets a list of equivalent entity types as inputs for the task.
	CreatingCommandName	Gets the name of a command that creates the container to be inserted in the task group.
	DeletingCommandName	Gets the name of a command that is invoked before deleting the container.
	UpdateCommandName	Gets the name of a command that updates the container's contents based on the upstream containers on which it depends.
	DataGroupUpdateCommandName	Gets the name of a command that updates the data groups of the task's container.

Class	Member	Description
	WillBePendingUponUpdateQueryName	Gets the name of a query that returns whether the task goes into the pending state upon update.
	WillBeUpdatedViaRsmQueryName	Gets the name of a query that returns whether the task is updated via the ANSYS Remote Solve Manager (RSM).
	RemoteSolvePermissionsQueryName	Gets the name of a query that returns whether the task can be part of a design point update via RSM.
	EditorInitiatedRemoteUpdateInfoQueryName	Gets the name of a query that returns all remote update information for this task that was initiated from within the add-in editor.
	RefreshCommandName	Gets the name of a command that refreshes the task's inputs from upstream containers.
	DataSourceListChangedCommandName	Gets the name of a command that handles input source changes for the task.
	TaskStatusQueryName	Gets the name of a query that returns the status of the task.
	DataGroupStatusQueryName	Gets the name of a query that returns the status of the data groups of the task's container.
	DuplicateTaskCommandName	Gets the name of a command that duplicates the entire task content.
	DuplicateUserDataCommandName	Gets the name of a command that duplicates only the user input data.
	AvailableTransferTasks	Gets the list of available transfer tasks that can be exposed from the task.
	ReportContentCreationCommandName	Gets the name of a command that provides general report content for the task.
	DesignPointReportContentCreationCommandName	Gets the name of a command that provides design point report content for the task.
	CleanCommandName	Gets the name of a command that cleans the container's heavyweight output data and deletes associated data files.
	ResetCommandName	Gets the name of a command that resets the container's internal data.
	AboutToResetCommandName	Gets the name of a command that prepares to reset the container's internal data. The task can stop resetting by throwing an exception.
	ComponentPropertyDataQueryName	Gets the name of a query that returns custom data to show in the task group Properties pane.

Class	Member	Description
	CanUseTransfer-DataQueryName	Gets the name of a query that returns whether a user-connection can be established between a transfer task and a data task.
	CanDuplicateQuery-Name	Gets the name of a query that returns whether a task created from the template can be duplicated.
	DuplicateGroup	Gets the group within which tasks can potentially be replaced or duplicated.
	ComponentImageName	Gets the name of the default image to use as the task icon.
	ValidationCodes	Gets the task's validation codes.
	DefaultTransferName	Gets the name given to the default outputs of the task if they are offered as part of a transfer selection popup.
	IncludeInPartialUpdate	Gets a value indicating whether to consider the task for a partial update.
	RefreshAfterPartialUpdate	Gets a value indicating whether to refresh the downstream task after a partial update.
	ShowOptionToAlwaysBePartOfDpUpdate	Gets a value indicating whether to give an option to always update the task as part of a design point update.
	GetEnvironmentForRsm-DpUpdateQueryName	Gets the name of a query that returns environment variables set as part of a design point update via RSM.
	GetRemoteUpdateMonitorFilesQueryName	Gets the name of a query that returns monitor files during a RSM design point update.
	SubTasks	Gets the list of subtasks.
	SubTaskStatusQuery-Name	Gets the name of a query that returns subtask states.
	IsDisabled	Gets a value indicating whether the template is a fake template for an orphaned task, where the normal template is not accessible. The task should be shown as disabled and no operations should be available.
	IsAim	Gets a value indicating whether the template is an AIM task template.
	InternalObject	Gets the internal component template represented by the task template.

Accessing Task Group APIs

Workbench task groups are exposed in the interface through Automation API wrappers in the namespace **Ansys.ACT.WorkBench.Automation.Workflows**. The project data model returns this wrapper type:

```
taskGroup = ExtAPI.DataModel.TaskGroups[0]
```

The task group wrapper does not store any local data. All data access and modification occur at member call time.

Note:

When a task group is accessed from an ACT script, an instance of **UserTaskGroup** is still used. A **UserTaskGroupTemplate** is now accessible from the property **Template**. All other task group functionality remains intact.

The following table lists task group APIs.

Class	Member	Description
ITaskGroup	Name	The internal task group name.
	Caption	The user-visible task group display text.
	Tasks	The tasks contained in the task group.
	InternalObject	The task group's internal object.
TaskGroup	TaskId	The internal Workbench system identifier.
	Caption	The Workbench UI-visible display text (system caption).
	Abbreviation	The Workbench system abbreviation.
	InternalObject	The Workbench system.
	AnalysisType	The Workbench system analysis type.
	DirectoryName	The Workbench system directory name.
	Physics	The list of Workbench system physics.
	Solver	The list of Workbench system solvers.
	Notes	The Workbench system notes.
	HeaderText	The Workbench system block title.
	IsParametric	Indicates whether the Workbench system operates solely on parameters (appears below the Parameter Set bar in the Workbench Project Schematic).
	Tasks	The list of tasks contained in this task group.
	Template	Gets the template from which the task group was created.
	GetTaskByName (name)	Gets the contained task by the supplied name.

Class	Member	Description
	Clean()	Clears generated data on all tasks in the task group.
	Copy()	Creates a new task group containing a copy of all the data in the task group.
	Delete()	Deletes the task group and all contained data from the project.
	Refresh()	Refreshes the input data for all tasks in the task group by reading changed data from upstream sources. No calculations or updates based on the new data are performed.
	Update()	Updates all tasks in the task group by refreshing the input from all upstream sources and then performing local calculation based on the current data.
	RecreateDeletedTasks()	Recreates new versions of any tasks that have been deleted from the task group.

Accessing Task Group Template APIs

Workbench system task group templates are exposed at the user level through Automation API wrappers in the namespace **Ansys.ACT.WorkBench.Automation.Workflows**. The project data model returns this wrapper type:

```
taskGroup = ExtAPI.DataModel.TaskGroups[0]
taskGroupTemplate = taskGroup.Template
```

The task group template wrapper does not store any local data. All data access and modification occur at member call time.

The following table lists task group APIs.

Class	Member	Description
TaskGroupTemplate	Name	Gets the task group template name.
	Category	Gets the toolbox category in which the template appears.
	TaskGroupType	Gets the task group type to display in the interface in the header of the task group block.
	TaskGroupTypeAbbreviation	Gets the task group type abbreviation, which is used to create the task group unique directory name and also serves as the task group object's base name.
	DisplayName	Gets the user-visible name for the type of task groups created from the template.
	ToolboxGroup	Gets the toolbox group in which the task group template should be displayed.
	ImageName	Gets the image associated with the task group template.
	Physics	Gets the physics associated with the task group template.

Class	Member	Description
	AnalysisType	Gets the analysis type associated with the task group template.
	TaskGroupHelpId	Gets the help ID associated with the task group template.
	ToolTipHelpId	Gets the help ID associated with the tool tip for the task group.
	IsParametric	Gets a value indicating whether the task group consumes parameters.
	Solver	Gets the solver associated with the task group template.
	TaskTemplates	Gets the task templates that make up the task group. Tasks are grouped into task groups. For the normal case with no groups, a single-element list is returned.
	TaskNames	Gets the names of the tasks to create from the templates in TaskTemplates .
	ExcludedTaskTemplates	Gets a list of task templates that are not permitted to be instantiated in the task group using such commands as InsertComponentBefore or InsertComponentAfter .
	RequiredTaskTemplates	Gets a list of task templates that cannot be deleted from the task group.
	Attributes	Gets the attributes to apply to the created task group.
	PostCreationSteps	Gets the list of operations to perform after the task group is created.
	Visible	Gets a value indicating whether the task group should appear in the toolbar.
	IsBeta	Gets a value indicating whether the task group visibility is tied to the beta option.
	IsDeletable	Gets a value indicating whether the task group can be deleted.
	InternalObject	Gets the internal system template represented by the task group template.

Accessing Parameter APIs

Workbench parameter objects are exposed through Automation API wrappers in the namespace **Ansys.ACT.WorkBench.Automation.Workflows**. These wrappers fully support simple parameter interactions:

```
task = ExtAPI.DataModel.Tasks[0]
p0 = task.Parameters[0]
value = p0.Value
newValue = value * 2
p0.Value = newValue
```

The following table lists the parameter APIs.

Class	Member	Description
IWBParameter	Name	The parameter name.
	Caption	The parameter display text.
	Value	The parameter value for the current design point.
	Usage	The parameter usage.
Parameter	Name	The parameter name.
	Caption	The parameter display text.
	Value	The parameter value for the current design point.
	Usage	The parameter usage.
	InternalObject	The Workbench parameter data reference.
ParameterUsage	Input	A parameter whose value is to be used by the data model.
	Output	A parameter whose value is either based on an expression or provided directly by the data model.
	Unknown	The parameter usage could not be determined.

Accessing Property APIs

Workbench property values are exposed through Automation API wrappers in the namespace **Ansys.ACT.WorkBench.Automation.Workflows**. These wrappers support simple property interactions:

```
task = ExtAPI.DataModel.Tasks[0]
p1 = task.Properties["PropertyOne"]
value = p1.Value
newValue = value * 2
p0.Value = newValue
```

The following table lists property APIs.

Class	Member	Description
IWBProperty	Name	The property name.
	Value	The property value.
Property	Name	The property name.
	Value	The property value.

Accessing State-Handling APIs

To control state handling, you can access state values from the **Ansys.ACT.Interfaces.Common.State** enumeration.

The following table lists state-handling APIs.

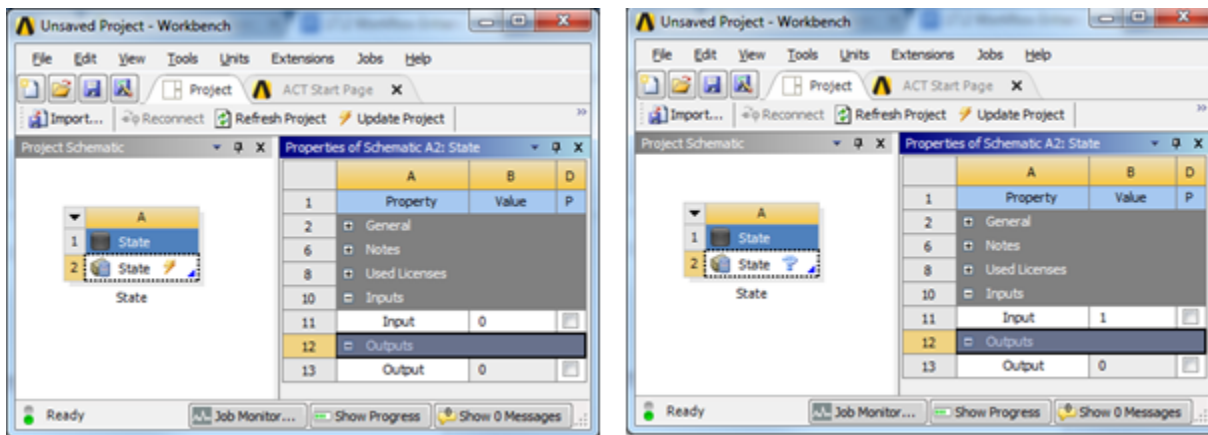
Class	Member	Description
Ansys.ACT.Interfaces.Common.State	UpToDate	The task is in an up-to-date state.
	RefreshRequired	The task requires a refresh to continue upstream changes.
	OutOfDate	The task is in an out-of-date state. This state is valid only in Mechanical.
	UpdateRequired	The task must be updated.
	Modified	The task has been modified. This state is valid only in Mechanical.
	Unfulfilled	Task requirements are unfulfilled.
	Disabled	The task is disabled.
	Error	The task is in error.
	EditRequired	Data entered for the task is invalid or incomplete and must be modified.
	Interrupted	Processing of the task is halted.
	UpstreamChangesPending	Updating of the task cannot be completed until previous tasks are updated.
	Unknown	The state of the task could not be determined.

When defining the state callback on tasks, you can return the framework-defined **ComponentState** instance. However, returning this instance is not recommended unless you must maintain backwards compatibility. Instead, return a two-entry list, where the first entry is an **Ansys.ACT.Interfaces.Common.State** enumeration value and the second entry is a quick-help string.

```
import clr
clr.AddReference('Ansys.ACT.Interfaces')
import Ansys.ACT.Interfaces

def status(task):
    if task.Properties['Inputs'].Properties['Input'].Value == 1:
        return [ Ansys.ACT.Interfaces.Common.State.Unfulfilled ,
                 'cannot enter the value of 1' ]
    else:
        return None #rely on the default framework-calculated state
```

When executed, this code sample yields the following result:



In addition, ACT provides a default state handler whenever the task does not provide a callback `<on-status>`. When any task property is invalid, ACT instructs the framework to use an unfulfilled state, as shown in the following examples of an XML file and IronPython script.

XML File:

```
<extension version="1" name="StateValidity">
  <guid shortid="StateValidity">
    69d0155b-e138-4841-a13a-del2238c83f2
  </guid>
  <script src="main.py" />
  <interface context="Project">
    <images>images</images>
  </interface>
  <workflow name="MyWorkflow" context="Project" version="1">
    <tasks>
      <task name="StateValidity" caption="State Validity" icon="Generic_cell" version="1">
        <callbacks>
          <onupdate>update</onupdate>
          <onreset>reset</onreset>
        </callbacks>
        <property name="Valid" caption="Property Check" control="integer" default="0" readonly="false">
          <callbacks>
            <isinvalid>valid</isinvalid>
          </callbacks>
        </property>
        <property name="Input" caption="Input" control="float" default="0.0" readonly="false" needupdate="true">
        </property>
        <property name="Output" caption="Output" control="float" default="0.0" readonly="true" visible="false">
        </property>
        <inputs>
          <input/>
        </inputs>
        <outputs/>
      </task>
    </tasks>
    <taskgroups>
      <taskgroup name="StateValidity" caption="State Validity" icon="Generic" category="" abbreviation="">
        <includeTask name="StateValidity" caption="State Validity"/>
      </taskgroup>
    </taskgroups>
  </workflow>
</extension>
```

IronPython Script:

```
import clr
clr.AddReference('Ansys.ACT.Interfaces')
import Ansys.ACT.Interfaces

def update(task):
    activeDir = task.ActiveDirectory
```

```

extensionDir = task.Extension.InstallDir
exeName = "ExampleAddinExternalSolver.exe"
solverPath = System.IO.Path.Combine(extensionDir, exeName)

inputValue = task.Properties["Inputs"].Properties["Input"].Value

inputFileName = "input.txt"
outputFileName = "output.txt"
dpInputFile = System.IO.Path.Combine(activeDir, inputFileName)
dpOutputFile = System.IO.Path.Combine(activeDir, outputFileName)

#write input file
f = open(dpInputFile, "w")
f.write('input='+inputValue.ToString(System.Globalization.NumberFormatInfo.InvariantInfo))
f.close()

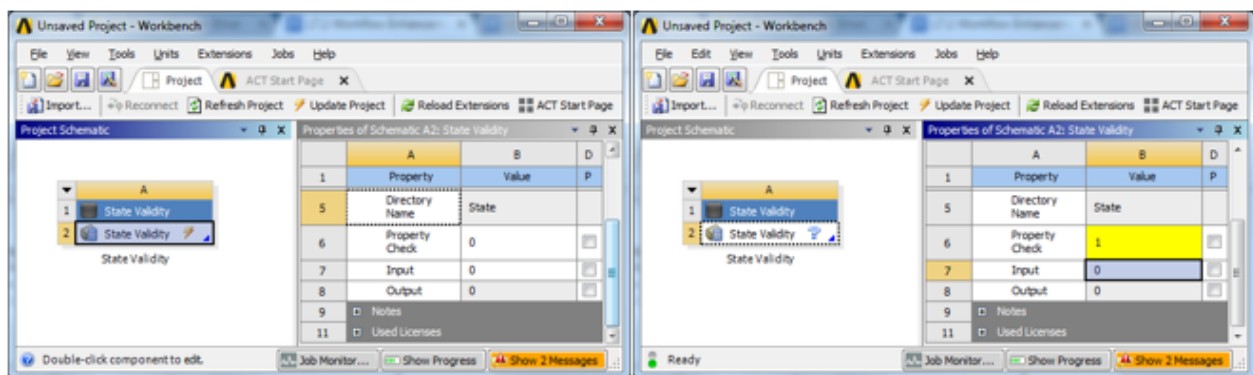
exitCode = ExtAPI.ProcessUtils.Start(solverPath, dpInputFile, dpOutputFile)
if exitCode != 0:
    raise Exception ('External solver failed!')
#read output file

outputValue = None
f = open(dpOutputFile, "r")
currLine = f.readline()
while currLine != "":
    valuePair = currLine.split('=')
    outputValue = System.Double.Parse(valuePair[1],System.Globalization.NumberFormatInfo.InvariantInfo)
    currLine = f.readline()
f.close()

if outputValue == None:
    raise Exception("Error in update - no output value detected!")
else:
    task.Properties["Output"].Value = outputValue
def reset(task):
    task.Properties["Input"].Value = 0
    task.Properties["Output"].Value = 0
def valid(entity, property):
    if property.Value == 1:
        return False
    else:
        return True

```

Result:



Accessing Project Reporting APIs

ACT fully wraps the framework's reporting API. To generalize APIs across all ANSYS products, interfaces have been defined. They can potentially be used to adopt reporting for other products outside of Workbench.

The following table lists project reporting APIs.

Class	Member	Description
IReport	AddChild(item)	Adds a child report item to the report.
	AddChild(item, index)	Adds a child report item to the report.
	RemoveChild(index)	Removes a child report item from the report.
	GetChild(index)	Obtains a child report item from the report.
	ChildCount	The number of children inside the report.
	FilePath	The report file path.
	ImageDirectoryPath	The image directory path for report images.
ProjectReport	AddReportImage(name)	Adds an image item to the report.
	InsertReportImage(name, index)	Inserts an image item into the report.
	AddReportLink(name)	Adds a link item to the report.
	InsertReportLink(name, index)	Inserts a link item into the report.
	AddReportSection(name)	Adds a section item to the report.
	InsertReportSection(name, index)	Inserts a section item into the report.
	AddReportText(name)	Adds a text item to the report.
	InsertReportText(name, index)	Inserts a text item into the report.
	AddReportTable(name)	Adds a table item to the report.
	AddReportTable(name, numberOfRows, numberOfColumns)	Adds a table item to the report, with the specified number of rows and columns.
	InsertReportTable(name, index)	Inserts a table item into the report.
IReportItem	Name	The name of the report item.
ProjectReportImage	ProjectReportImage(name)	Constructor.
	ProjectReportImage(name, sourceLocation)	Constructor.
	SourceLocation	The image location (file path).
ProjectReportLink	ProjectReportLink(name)	Constructor.
	ProjectReportLink(name, resourceLocation)	Constructor.
	ResourceLocation	The linked path.

Class	Member	Description
ProjectReportSection	ProjectReportSection()	Constructor.
	ProjectReportSection(name)	Constructor.
	AddChild(item)	Adds a child report item to the report section.
	AddChild(item, index)	Adds a child report item to the report section at the specified index.
	RemoveChild(index)	Removes a child report item from the section at the given index.
	GetChild(index)	Retrieves a child report item from the section at the given index.
	ChildCount	The number of children inside the report section.
	AddReportImage(name)	Adds an image item to the report.
	InsertReportImage(name, index)	Inserts an image item into the report section.
	AddReportLink(name)	Adds a link item to the report section.
	InsertReportLink(name, index)	Inserts a link item into the report section.
	AddReportSection(name)	Adds a section item to the report section.
	InsertReportSection(name, index)	Inserts a section item into the report section.
	AddReportText(name)	Adds a text item to the report section.
	InsertReportText(name, index)	Inserts a text item into the report section.
	AddReportTable(name)	Adds a table item to the report section.
	AddReportTable(name, numberOfRows, numberOfColumns)	Adds a table item to the report section, with the specified number of rows and columns.
	InsertReportTable(name, index)	Inserts a table item into the report section.
ProjectReportTable	ProjectReportTable(name)	Constructor.
	ProjectReportTable(name, numRows, numColumns)	Constructor.
	RowCount	The number of rows in the table.
	ColumnCount	The number of columns in the table.

Class	Member	Description
	GetColumnName(index)	Retrieves the name for a column at the specified index.
	SetColumnName(index)	Sets the name for a column at the specified index.
	AddColumn(columnName)	Adds an empty column to the table using the supplied column name.
	AddColumn(columnName, index)	Adds an empty column to the table at the specified index using the supplied column name.
	AddColumn(columnName, index, columnValues)	Adds a column to the table at the specified index using the supplied column name and populated with the supplied values.
	RemoveColumn(index)	Removes a column at the specified index.
	AddRow(rowValues)	Adds a new row of values to the table.
	AddRow(rowValues, index)	Adds a new row of values to the table at the specified index.
	RemoveRow(index)	Removes a row from the table at the specified index.
	GetValue(rowIndex, columnIndex)	Retrieves a value from the table at the specified row-and-column coordinates.
	SetValue(rowIndex, columnIndex, value)	Sets a report table value in the table at the supplied row-and-column coordinates.
	SetValue(rowIndex, columnIndex, content)	Sets a string value in the table at the supplied row-and-column coordinates.
	SetRowBackgroundColor(index, color)	Sets the background color for the row at the specified index.
	GetRowBackgroundColor(index)	Retrieves the current row background color for the row at the supplied index.
	SetRowForegroundColor(index, color)	Sets the foreground color for the row at the specified index.
	GetRowForegroundColor(index)	Retrieves the current row foreground color for the row at the supplied index.
	ShowHeaders	Indicates whether or not the column headers should be displayed.

Class	Member	Description
	IsCollapsible	Indicates whether or not the table is collapsible.
ProjectReportTableValue	ProjectReportTableValue(content)	Constructor.
	ProjectReportTableValue(content, iconSourceLocation, iconAlternateText)	Constructor.
	Content	The table value string.
	IconSourceLocation	The cell icon source path.
	IconAlternateText	The cell icon alternate text to use when the icon cannot be located.
ProjectReportText	ProjectReportText(text)	Constructor.
	ProjectReportText(name, text)	Constructor.
	Text	Gets or sets the report item's text.

Consider the following code sample:

```
import clr
clr.AddReference("Ansys.ACT.WorkBench")
import Ansys.ACT.WorkBench

def report(task, report):
    section = report.AddReportSection("My Custom ACT Task Report Content")
    text = section.AddReportText("TestText")
    text.Text = "Sample text from the data squares component"
    link = section.AddReportLink("TestLink")
    link.ResourceLocation = r"http://www.ansys.com"
    extDir = ExtAPI.ExtensionManager.CurrentExtension.InstallDir
    imgDir = System.IO.Path.Combine(extDir, "images")
    imgName = "logo-ansys.jpg"
    imgPath = System.IO.Path.Combine(imgDir, imgName)
    reportImgDir = report.ImageDirectoryPath
    destImgPath = System.IO.File.Copy(imgPath, System.IO.Path.Combine(reportImgDir, imgName))
    img = section.AddReportImage("TestImage")
    img.SourceLocation = imgName

    rowCount = 5
    colCount = 3
    table = section.AddReportTable("TestTable", rowCount, colCount)
    for i in range(0, rowCount):
        evenRow = (i % 2 == 0)
        rowVals = []
        for j in range(0, colCount):
            table.SetValue(i, j, str(j + i*colCount))
        if evenRow:
            table.SetRowBackgroundColor(i, "Red")
            table.SetRowForegroundColor(i, "Blue")
        else:
            table.SetRowBackgroundColor(i, "Blue")
            table.SetRowForegroundColor(i, "Red")
    table.SetColumnName(0, "Col 1")
    table.SetColumnName(1, "Col 2")
    table.SetColumnName(2, "Col 3")
```

When executed, this code sample generates the following report:

Reporting


My Custom ACT Task Report Content

TestText

Sample text from the data squares component

TestLink

TestImage



TestTable

Col 1	Col 2	Col 3
0	1	2
3	4	5
6	7	8
9	10	11
12	13	14

Workbench Feature Creation

In addition to supporting the [common feature creation](#) capabilities described in the *ANSYS ACT Developer's Guide*, Workbench supports product-specific feature creation capabilities. The following topics describe how to implement a custom interface operation for a task, use global callbacks to implement a custom process or action before or after a Workbench **Project Schematic** operation, and define ACT-based properties as parameters in Workbench:

[Custom User-Specified Interface Operation](#)

[Global Workflow Callbacks](#)

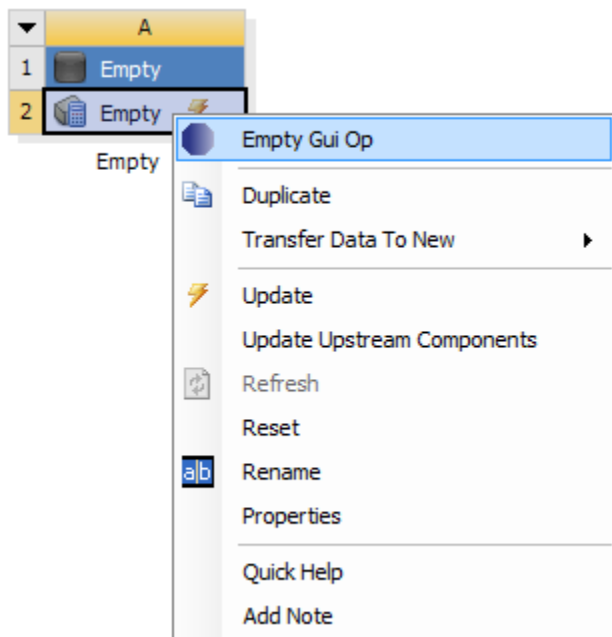
[ACT-Based Property Parameterization in Workbench](#)

Note:

Workbench requires PNG files for the images to display as toolbar buttons.

Custom User-Specified Interface Operation

The supplied extension **EmptyGUI** implements a custom interface operation for a Workbench task. Specifically, it adds a custom context menu. This functionality is valuable when you require menu entries beyond the default **Edit** menu created by the definition of the callback `<onedit>`.



Creating the Extension for Custom Interface Operations

The file `EmptyGUI.xml` follows.

```
<extension version="1" name="EmptyGUI">
  <guid shortid="EmptyGUI">69d0095b-e138-4841-a13a-de12238c83f6</guid>
  <script src="empty_gui.py" />
  <interface context="Project">
    <images>images</images>
  </interface>
  <workflow name="wf3" context="Project" version="1">
    <tasks>
      <task name="Empty" caption="Empty" icon="Generic_cell" version="1">
        <callbacks>
          <onupdate>update</onupdate>
        </callbacks>
        <inputs>
          <input/>
        </inputs>
        <outputs/>
        <contextmenus>
          <entry name="Empty Gui Op" type="ContextMenuEntry" priority="1.0" icon="default_op">
            <callbacks>
              <onclick>click</onclick>
            </callbacks>
          </entry>
        </contextmenus>
      </task>
    </tasks>
    <taskgroups>
      <taskgroup name="Empty" caption="Empty" icon="Generic" category="ACT Custom Workflows" abbreviation=">
        <includeTask name="Empty" caption="Empty"/>
      </taskgroup>
    </taskgroups>
  </workflow>
</extension>
```

This XML file performs the following actions:

- References the IronPython script `empty_gui.py`.
- Defines a single task in the element `<tasks>`. Within this element, a single context menu is defined in the element `<contextmenus>`. The callback `<onclick>` is defined, as required.
- Defines the inputs and outputs in the elements `<inputs>` and `<outputs>`. Note that an empty input and an output are defined.
- Defines a single task group with a single task in the element `<taskgroups>`.

Defining Functions for Custom Interface Operations

The IronPython script `empty_gui.py` defines the functions that the task executes from the context menu and update callbacks. Because the XML file has both the callbacks `<onupdate>` and `<onclick>`, the update and click methods are defined in the script.

```
import clr
clr.AddReference("Ans.UI.Toolkit")
clr.AddReference("Ans.UI.Toolkit.Base")
import Ansys.UI.Toolkit

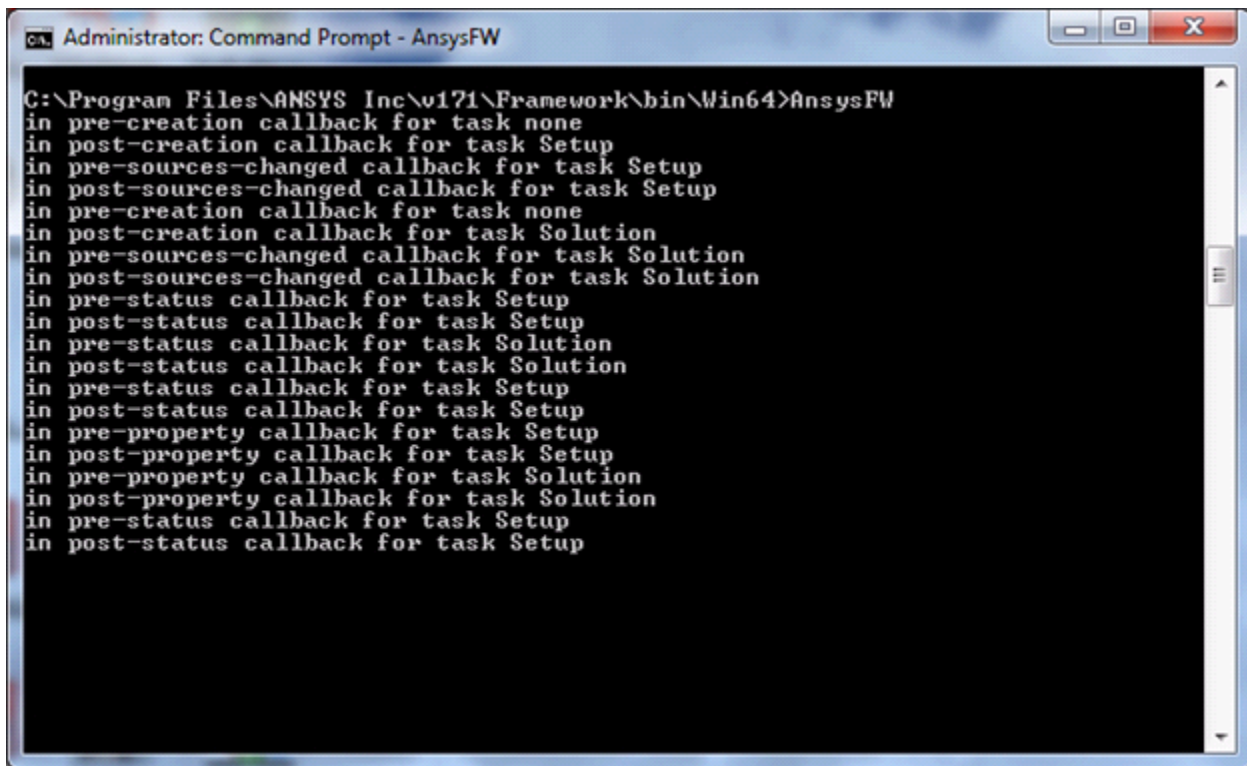
def click(task):
    Ansys.UI.Toolkit.MessageBox.Show("Empty Test!")
```

```
def update(task):
    ExtAPI.Log.WriteMessage('empty update')
```

Global Workflow Callbacks

The supplied extension **WorkflowCallbacksDemo** demonstrates the use of global callbacks to implement a custom process or action before or after a Workbench **Project Schematic** operation. Specifically, it defines pre- and post-operation callbacks for each available operation. Each task, or argument passed to the invoked method, defines an action: writing a message in the command line window either before or after execution of the operation.

This example also shows how to specify that a callback method (in this case, **onAfterUpdate**) is processed only for a specific task or template.



Creating the Extension for Global Callbacks

The file `WorkflowCallbacksDemo.xml` follows.

```
<extension version="1" name="WorkflowCallbacksDemo">
  <guid shortid="WorkflowCallbacksDemo">96d0195b-e138-4841-a13a-de12238c83f2</guid>
  <script src="main.py" />
  <interface context="Project">
    <images>images</images>
  </interface>
  <workflow name="WorkflowDemo1" context="Project" version="1">
    <callbacks>
      <onbeforetaskreset>onBeforeReset</onbeforetaskreset>
      <onaftertaskreset>onAfterReset</onaftertaskreset>
      <onbeforetaskrefresh>onBeforeRefresh</onbeforetaskrefresh>
      <onaftertaskrefresh>onAfterRefresh</onaftertaskrefresh>
      <onbeforetaskupdate>onBeforeUpdate</onbeforetaskupdate>
```

```

<onaftertaskupdate>onAfterUpdate</onaftertaskupdate>
<onbeforetaskduplicate>onBeforeDuplicate</onbeforetaskduplicate>
<onaftertaskduplicate>onAfterDuplicate</onaftertaskduplicate>
<onbeforetasksourceschanged>onBeforeSourcesChanged</onbeforetasksourceschanged>
<onaftertasksourceschanged>onAfterSourcesChanged</onaftertasksourceschanged>
<onbeforetaskcreation>onBeforeCreate</onbeforetaskcreation>
<onaftertaskcreation>onAfterCreate</onaftertaskcreation>
<onbeforetaskdeletion>onBeforeDelete</onbeforetaskdeletion>
<onaftertaskdeletion>onAfterDelete</onaftertaskdeletion>
<onbeforetaskcanusetransfer>onBeforeCanUseTransfer</onbeforetaskcanusetransfer>
<onaftertaskcanusetransfer>onAfterCanUseTransfer</onaftertaskcanusetransfer>
<onbeforetaskcanduplicate>onBeforeCanDuplicate</onbeforetaskcanduplicate>
<onaftertaskcanduplicate>onAfterCanDuplicate</onaftertaskcanduplicate>
<onbeforetaskstatus>onBeforeStatus</onbeforetaskstatus>
<onaftertaskstatus>onAfterStatus</onaftertaskstatus>
<onbeforetaskpropertyretrieval>onBeforePropertyRetrieval</onbeforetaskpropertyretrieval>
<onaftertaskpropertyretrieval>onAfterPropertyRetrieval</onaftertaskpropertyretrieval>
</callbacks>
</workflow>
</extension>

```

This XML file performs the following actions:

- References the IronPython script `main.py`.
- For the element `<interface>`, specifies that the attribute `context` is set to **Project** so that the extension is executed from the **Project Schematic**.
- For the element `<workflow>`:
 - Defines the attributes `name` and `context`.
 - Defines pre- and post-operation global callbacks for each available **Project Schematic** operation.

Defining Functions for Global Callbacks

The IronPython script `main.py` defines the methods invoked by the global callbacks.

```

def onBeforeReset(task):
    msg = getPrintMessage('pre-reset', task)
    print msg
def onAfterReset(task):
    msg = getPrintMessage('post-reset', task)
    print msg
def onBeforeRefresh(task):
    msg = getPrintMessage('pre-refresh', task)
    print msg
def onAfterRefresh(task):
    msg = getPrintMessage('post-refresh', task)
    print msg
def onBeforeUpdate(task):
    if task.Name == "Engineering Data":
        msg = getPrintMessage('post-update', task)
        print msg
    else:
        print("ignored")
def onAfterUpdate(task):
    if task.Name == "Engineering Data":
        msg = getPrintMessage('post-update', task)
        print msg
    else:
        print("ignored")
def onBeforeDuplicate(task):
    msg = getPrintMessage('pre-duplicate', task)

```

```

    print msg
def onAfterDuplicate(task):
    msg = getPrintMessage('post-duplicate', task)
    print msg
def onBeforeSourcesChanged(task):
    msg = getPrintMessage('pre-sources-changed', task)
    print msg
def onAfterSourcesChanged(task):
    msg = getPrintMessage('post-sources-changed', task)
    print msg
def onBeforeCreate(task):
    msg = getPrintMessage('pre-creation', task)
    print msg
def onAfterCreate(task):
    msg = getPrintMessage('post-creation', task)
    print msg
def onBeforeDelete(task):
    msg = getPrintMessage('pre-deletion', task)
    print msg
def onAfterDelete(task):
    msg = getPrintMessage('post-deletion', task)
    print msg
def onBeforeCanUseTransfer(sourceTask, targetTask):
    msg = 'in pre-can-use-transfer with source task ' + sourceTask.Name +
        ' and target task ' + targetTask.Name
    print msg
def onAfterCanUseTransfer(sourceTask, targetTask):
    msg = 'in post-can-use-transfer with source task ' + sourceTask.Name +
        ' and target task ' + targetTask.Name
    print msg
def onBeforeCanDuplicate():
    msg = getPrintMessage('pre-can-use-transfer', None)
    print msg
def onAfterCanDuplicate():
    msg = getPrintMessage('post-can-use-transfer', None)
    print msg
def onBeforeStatus(task):
    msg = getPrintMessage('pre-status', task)
    print msg
def onAfterStatus(task):
    msg = getPrintMessage('post-status', task)
    print msg
def onBeforePropertyRetrieval(task):
    msg = getPrintMessage('pre-property', task)
    print msg
def onAfterPropertyRetrieval(task):
    msg = getPrintMessage('post-property', task)
    print msg
def getPrintMessage(msg, task):
    taskName = 'none'
    if task != None:
        taskName = task.Name
    return 'in ' + msg + ' callback for task ' + taskName

```

This script performs the following actions:

- Defines 23 methods:
 - One method for each of the 22 global callback defined in the XML extension definition file.
 - One method to format a message to write to the command line window.
- Specifies (via the method arguments) that the methods **onBeforeUpdate** and **onAfterUpdate** should be executed only for updates of **Engineering Data** tasks.

ACT-Based Property Parameterization in Workbench

In Workbench, you can define ACT-based task properties as input parameters, output parameters, or non-parameterized values. This topic describes how to implement input and output parameters on ACT objects within the Workbench **Project Schematic**.

To define either an input parameter or an output parameter in your **Project Schematic** workflow, add the attribute **isparameter** to the XML extension definition of the task-level property that you want to parameterize.

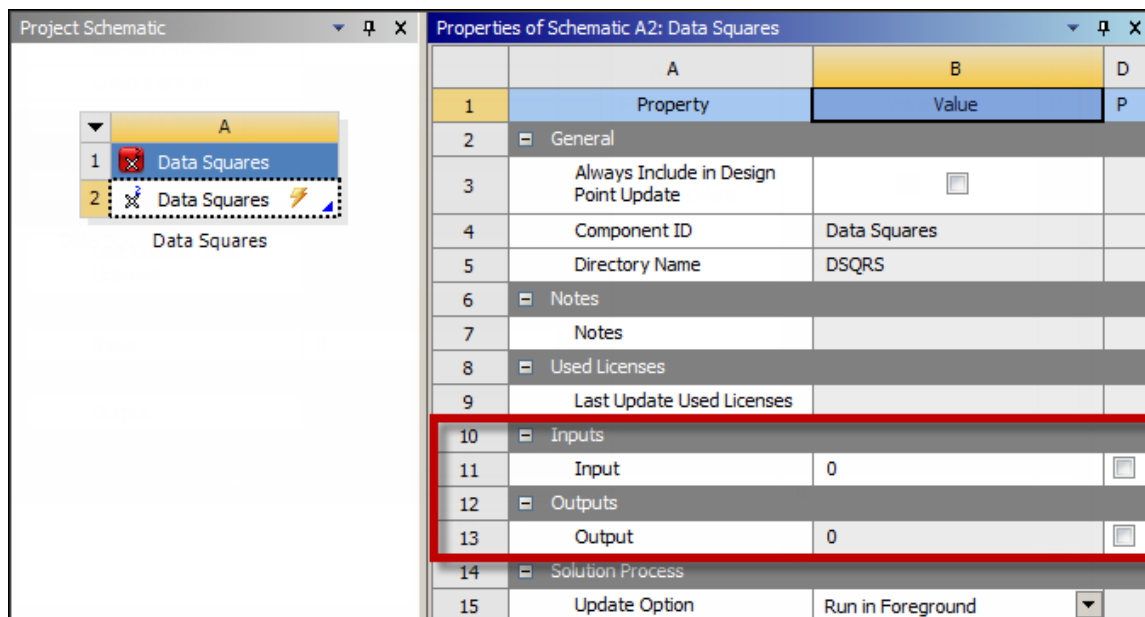
To demonstrate, the [supplied](#) extension **DataSquares** is used. The following code creates and parameterizes a task-level input property named **Input**. Because it is an input, the attribute **readonly** is set to **False**. The property also has the attribute **control** set to **integer**.

```
...
<propertygroup name="Inputs">
  <property name="Input" caption="Input" control="integer" default="0" readonly="False" needupdate="true" visible=
</propertygroup>
...
```

In the same way, the following code creates and enables parametrization for a task-level output property named **Output**. Because it is an output property, the attribute **readonly** is set to **true**. The property also has the attribute **control** set to **integer**.

```
...
<propertygroup name="Outputs">
  <property name="Output" caption="Output" control="integer" default="0" readonly="True" visible="True" persistent=
</propertygroup>
...
```

A custom system named **DataSquares** is created in the **Project Schematic**. The new properties are shown in the custom **Inputs** and **Outputs** sections exposed in the task properties. These custom sections are defined by the task groups.



When you select the check box for parametrizing the two new properties, they are added to the **Parameter Set** bar.

The supplied extensions **Squares** and **DataSquares** demonstrate how to use custom task properties to integrate lightweight external applications into your Workbench workflow. For more information, see:

- [External Application Integration with Parameter Definition \(p. 88\)](#)
- [External Application Integration with Custom Data and Remote Job Execution \(p. 92\)](#)

Simulation Workflow Integration

ACT allows you to integrate external applications and custom processes into the Workbench workflow. Features exposed by ACT also enable you to perform automation and customization activities, such as creating new systems to facilitate interaction with the Workbench **Project Schematic**.

The following sections provide information about using custom ACT workflows in Workbench:

[Custom Task Group and Task Exposure in the Project Schematic](#)

[ACT Workflow Designer](#)

[Global Callbacks](#)

[Data Transfers](#)

[Progress Monitoring](#)

[Process Utilities](#)

[Manual Creation of a Custom Workflow](#)

[XML Extension Definition for a Custom Workflow](#)

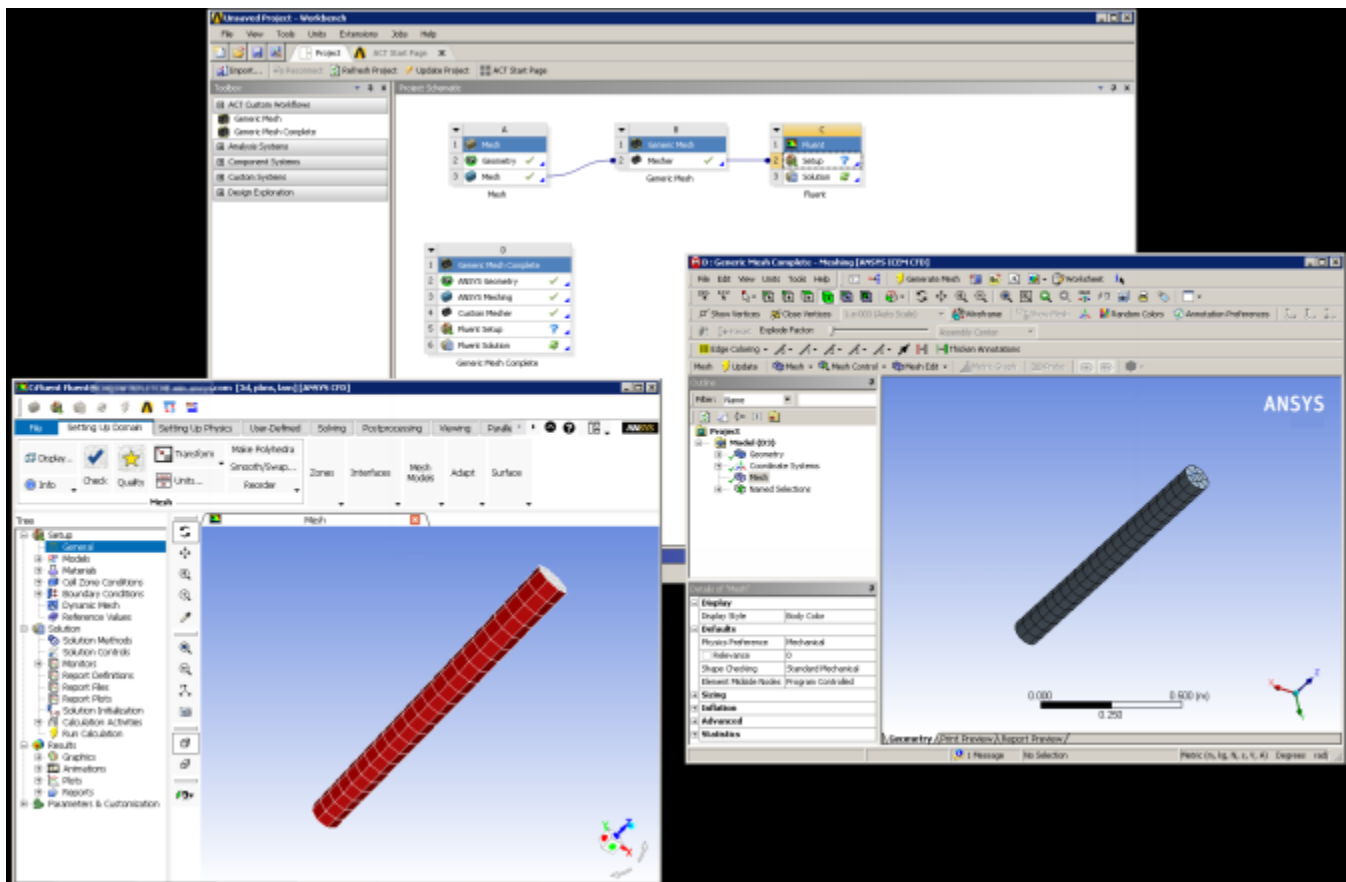
[IronPython Script for a Custom Workflow](#)

Custom Task Group and Task Exposure in the Project Schematic

You can use ACT to create custom task groups and custom tasks in the Workbench **Project Schematic**. Workbench refers to a task group as a *system* and to a task as a *cell*.

- Task groups contain tasks. ACT-defined task groups are exposed as custom systems in the Workbench **Toolbox** so that you can add them to the **Project Schematic** in the same way as you add an ANSYS-installed system.
- The tasks in a custom task group can include both ACT-defined tasks and ANSYS-installed tasks.

The following figure shows a **Generic Mesh** task group that takes an upstream mesh and passes it to a downstream **Fluent** task group. For more information, see [Generic Mesh Transfer \(p. 99\)](#).



ACT Workflow Designer

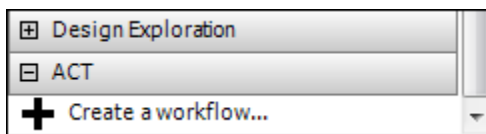
ACT has supported extending Workbench with custom workflows since Revision 17.2. However, to introduce an external application to Workbench's drag-and-drop simulation environment, you had to [manually create an ACT workflow extension \(p. 52\)](#) that defined your custom task group and custom tasks.

While you can still create ACT workflow extensions manually, you can also now use the **ACT Workflow Designer**. This lightweight integration automates workflow setup, saving you from having to create the extension's XML definition file and IronPython scripts directly.

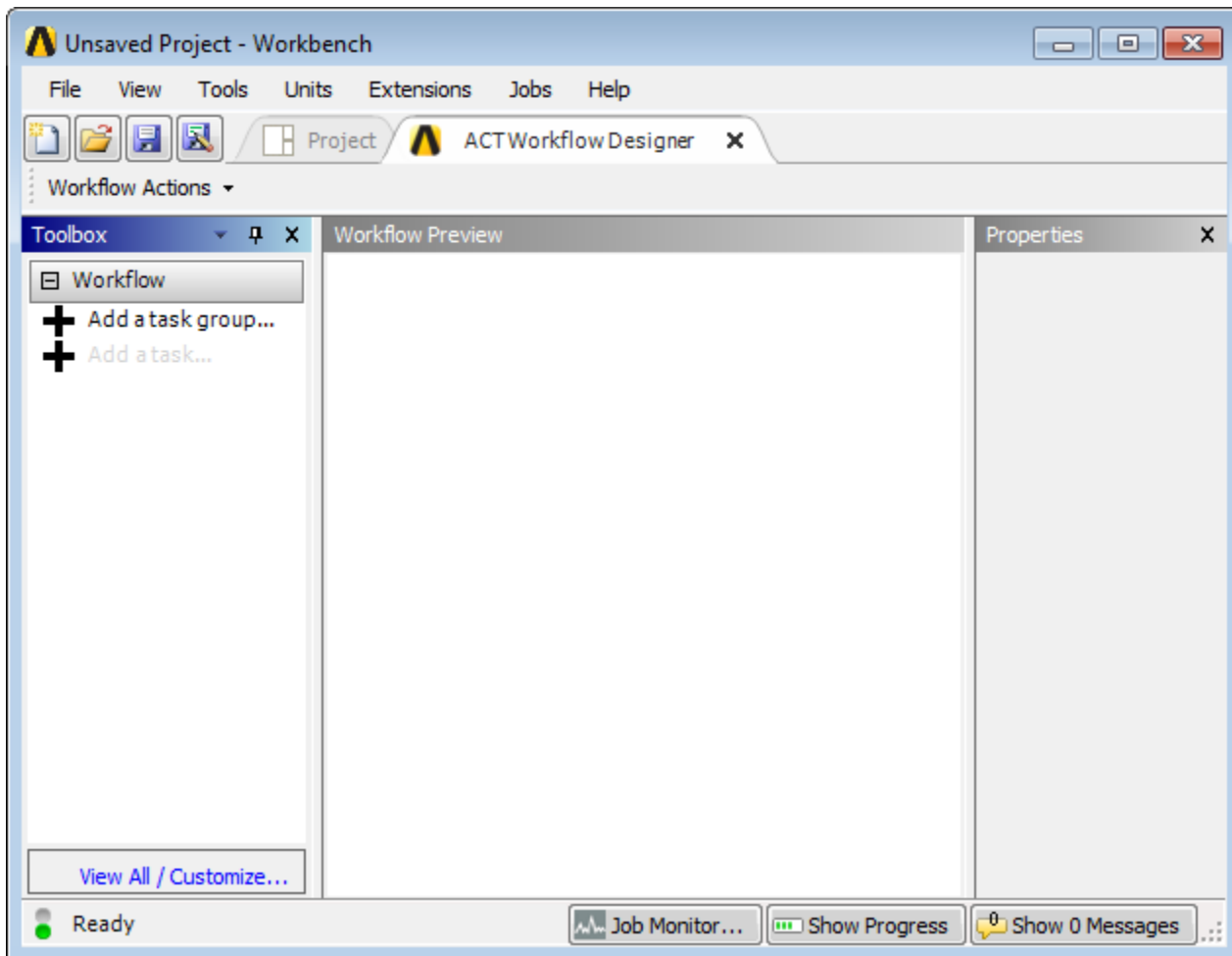
This section demonstrates how easy it is to use the **ACT Workflow Designer** to define a custom task group and custom tasks. [Global Callbacks \(p. 46\)](#) and subsequent sections then provide technical information and the process for manually creating custom workflows.

ACT Workflow Designer Startup

To start the **ACT Workflow Designer**, in the Workbench **Toolbox** under **ACT**, double-click **Create a workflow...**



The **ACT Workflow Designer** opens on a new Workbench tab. It offers an interactive, preview-based approach to task group and task definition. When you use the **ACT Workflow Designer**, you do not have to exit Workbench to create a workflow extension and then restart Workbench and load this extension.

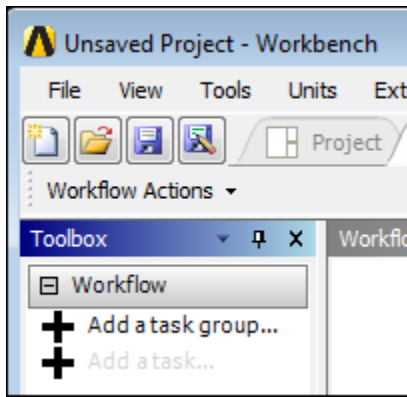


The **ACT Workflow Designer** consists of three panes:

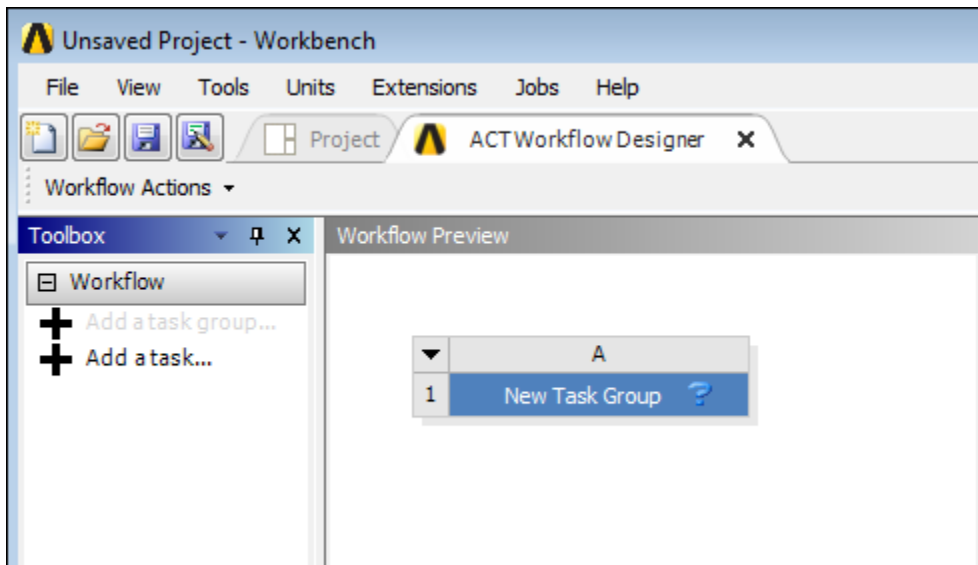
- **Toolbox**
- **Workflow Preview**
- **Properties**

Toolbox Pane

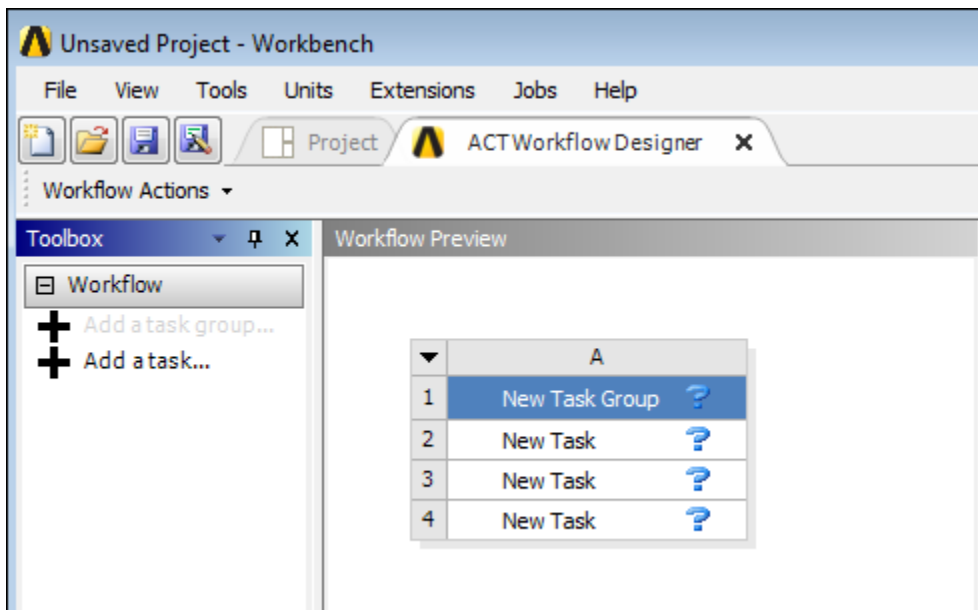
In the **Toolbox** pane, two entries exist under **Workflow**: **Add a task group** and **Add a task**. You click these entries to build your custom workflow in the **Workflow Preview** workspace. If an entry is disabled (grayed out), the item is either currently invalid or already exists in the workflow. For example, the **Add a task** entry is disabled until after a task group is inserted. Clicking a disabled entry results in no action.



Double-clicking **Add a task group** adds a **New Task Group** cell to the workspace. Only one task group instance at a time is supported presently. Once you add the first and only instance of the task group, **Add a task group** becomes disabled and **Add a task** becomes enabled.



Double-clicking **Add a task** adds a **New Task** cell to the end of the task group. You can add any number of tasks to the task group.



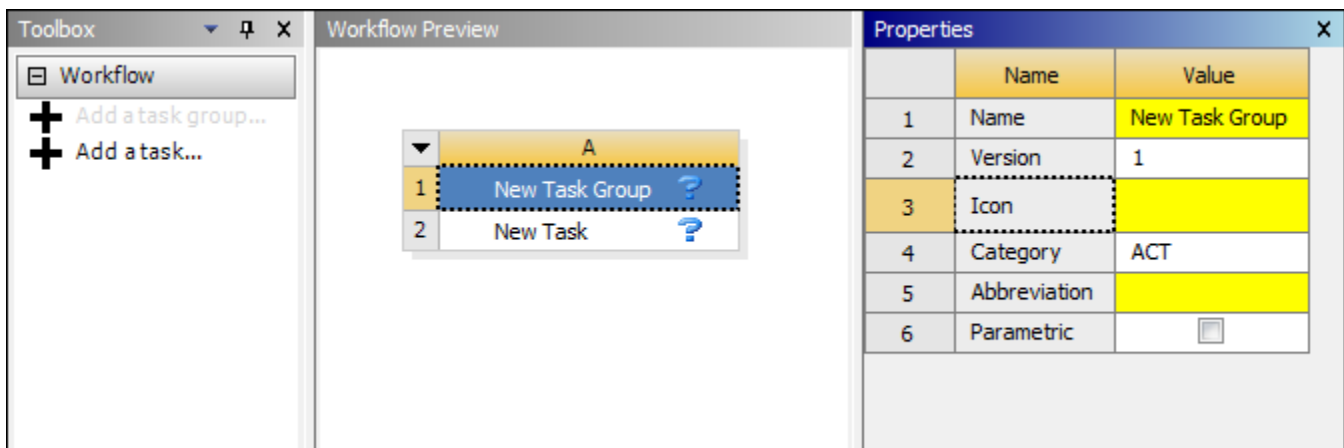
Workflow Preview Pane

The **Workflow Preview** pane provides a real-time visual display of the workflow under construction. You preview the task group and tasks as they will appear inside Workbench. The layout, text, and icons are a true representation of what the final workflow will look like.

Each cell displays a state icon on the right side. This icon indicates whether the cell has met minimum publishing requirements. A solid blue question mark indicates that the cell contains the data that must be completed before publishing. A green check mark indicates a cell contains all the data required for publishing.

Properties Pane

The **Properties** pane behaves in much the same way as the standard Workbench **Properties** pane. It displays a table of property names and values for the cell selected in the **Workflow Preview** workspace. If no cell is selected, no property data is shown. Property values requiring your attention are highlighted in yellow. Once the requirement for a property value is met, the highlight is removed.



As you edit property values, the **ACT Workflow Designer** refreshes all panes to provide real-time design verification.

Defining Properties for Workflow Cells

For each workflow cell, you must define the minimum properties required for publishing. After all cells have green check marks for their state icons, you can publish the workflow.

1. In the **Workflow Preview** workspace, select the task group cell.

The **Properties** pane displays the properties for the task group.

2. Define task group properties, referring to the descriptions below if necessary.
3. For each task cell in the task group, do the following:

- a. In the **Workflow Preview** workspace, select the task cell.

The **Properties** pane displays the properties for the task.

- b. Define task properties, referring to the descriptions below if necessary.

Task Group Properties

Descriptions follow for task group properties:

- **Name:** Name of the task group, which displays in the Workbench **Toolbox** under the group chosen for **Category** once the workflow is published.
- **Version:** Release number for this iteration of the task group. The default is **1**.
- **Icon:** Image file to display as the icon for the task group.
- **Category:** Group under which to place the workflow in the Workbench **Toolbox**. The default is **ACT**.
- **Abbreviation:** Workbench system abbreviation.
- **Parametric:** Indicates whether the task group operates only on design points. As such, the task group is added below the Workbench **Project Schematic**. This focus on parameters enables you to incorporate DesignXplorer-like functionality.

Task Properties

Descriptions follow for task properties:

- **Name:** Name of the task.
- **Version:** Release number for this iteration of the task. The default is **1**.
- **Icon:** Image file to display as the icon for the task.

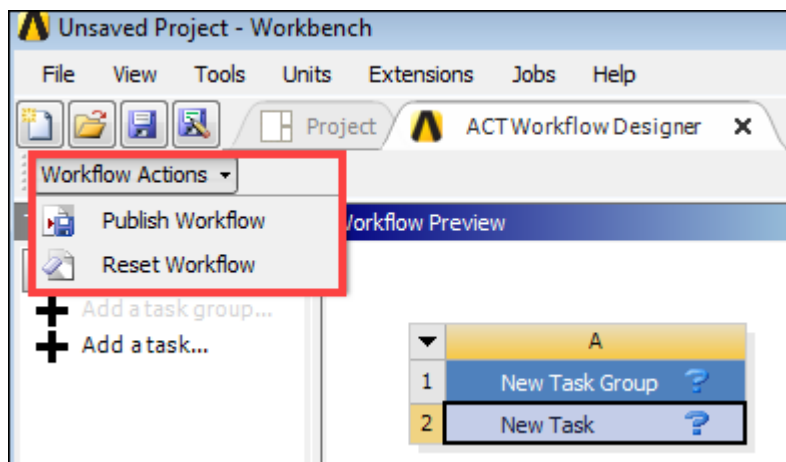
- **Application Path:** Path and name for the file to execute, which can be an EXE file for an external application or any file with which a system-installed program has been associated.

Note:

The application path is a fully qualified path. If you intend to share this project, use an environment variable in the application path so that others can then set this environment variable to the location on their machines where the executable file for the external application resides. For example, assume that the fully qualified path is `E:\WorkflowDesignerInputFiles\ExampleAddinExternalSolver.exe`. By changing it to `%ExternalSolverExecutable%\ExampleAddinExternalSolver.exe`, you allow users to set the path to the executable file in the environment variable `%ExternalSolverExecutable%`.

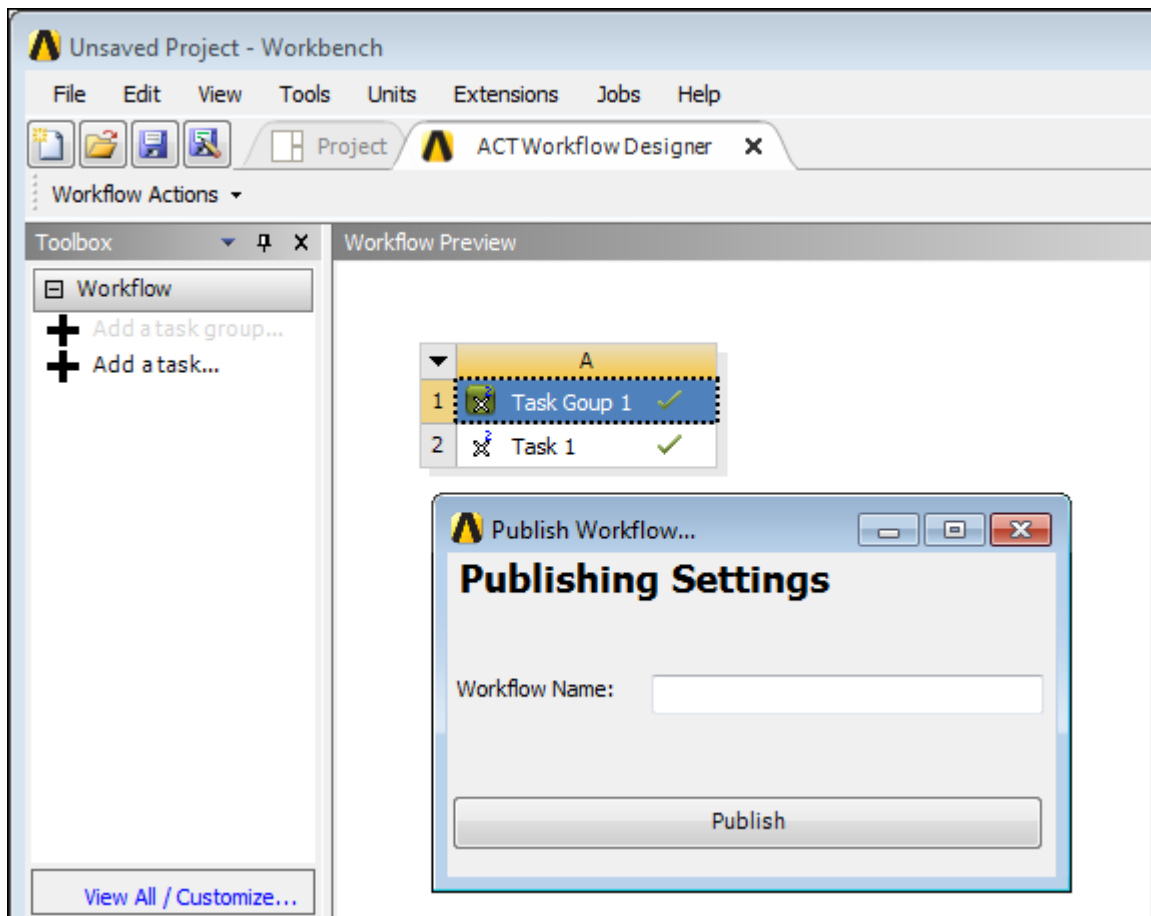
Publishing and Resetting Workflows

In the **ACT Workflow Designer**, the **Workflow Actions** menu provides two options: **Publish Workflow** and **Reset Workflow**.



Publishing the Workflow

Publishing a workflow imports it directly into the running Workbench session. Once all cells have green check marks, select **Workflow Actions** → **Publish Workflow**. In the **Publishing Settings** dialog box that opens, enter a workflow name and click **Publish**.

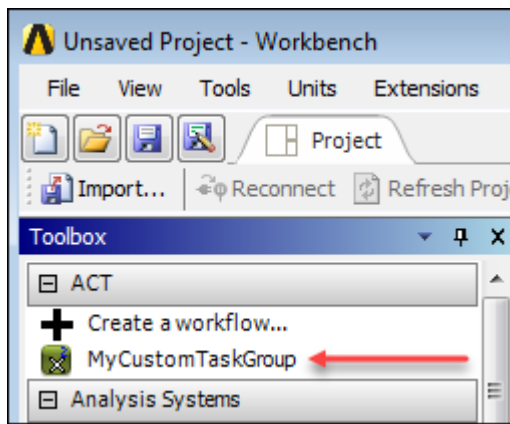


During the publishing process, the **ACT Workflow Designer** collects all workflow data, creates the scripted ACT extension, and then loads this extension in Workbench. The extension is created here: `C:\Users\YourUserName\AppData\Local\Roaming\Ansys\vXXX\ACT\extensions`, where `vXXX` is the active ANSYS version.

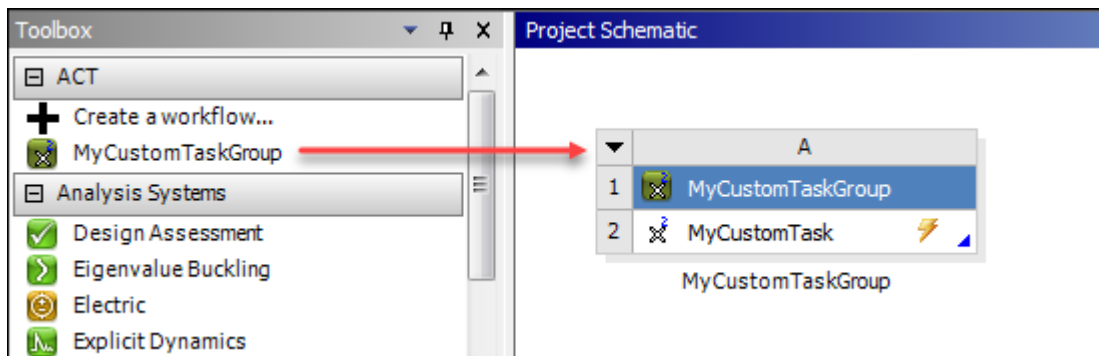
The **ACT Workflow Designer** produces the following extension members by default:

- Extension XML definition file
- Extension folder
 - **Images** folder, which contains any icon files specified
 - **main.py** script

After successfully publishing the workflow, the **ACT Workflow Designer** closes, returning you to the Workbench **Project Schematic**. Under the Workbench **Toolbox** category that you specified, an entry now exists for the new, fully operational workflow. When the category is **ACT** (default), the new workflow entry is added below the entry for creating a workflow.



To begin simulation activity, you drag the workflow entry from the Workbench **Toolbox** and drop it on the **Project Schematic**. You can also double-click this Workbench **Toolbox** entry to add it to the **Project Schematic**.



While the update icon displays on task cells, attempting to update a cell starts the external application, which generally fails because [input and output properties](#) (p. 40) for this external application are not yet defined.

You can share the scripted extension that publishing the workflow has created or use the [Binary Extension Builder](#) to produce a binary version (WBEX file) to deploy on the [ANSYS Store](#).

Tip:

If changes to this custom workflow are necessary, you can directly edit the XML file and IronPython script for the extension.

Resetting the Workflow

If before publishing a workflow, you select **Workflow Actions** → **Rest Workflow**, the **ACT Workflow Designer** clears the **Workflow Preview** workspace, removing the added task group and tasks.

Caution:

If you reset the workflow or close the **ACT Workflow Designer**, you lose all work. Clicking **Create a workflow** in the Workbench **Toolbox** under **ACT** to restart the **ACT Workflow Designer** activates a new workspace. You can save your workflow

design only by publishing it. Because publishing a workflow with incomplete data can result in an extension that cannot be loaded, you should complete the entire design before you publish the workflow and close the **ACT Workflow Designer**.

Defining Task Input and Output Properties

After you add a custom workflow to the **Project Schematic**, for each task cell, you define input and output files, the values to display as parameters in the Workbench **Parameter Set** bar, and command line arguments.

1. In the **Project Schematic**, right-click the task cell in the custom workflow and select **Properties**. The **Properties** pane displays. Properties that require values are highlighted in yellow.

Project Schematic

A


1 MyCustomTaskGroup

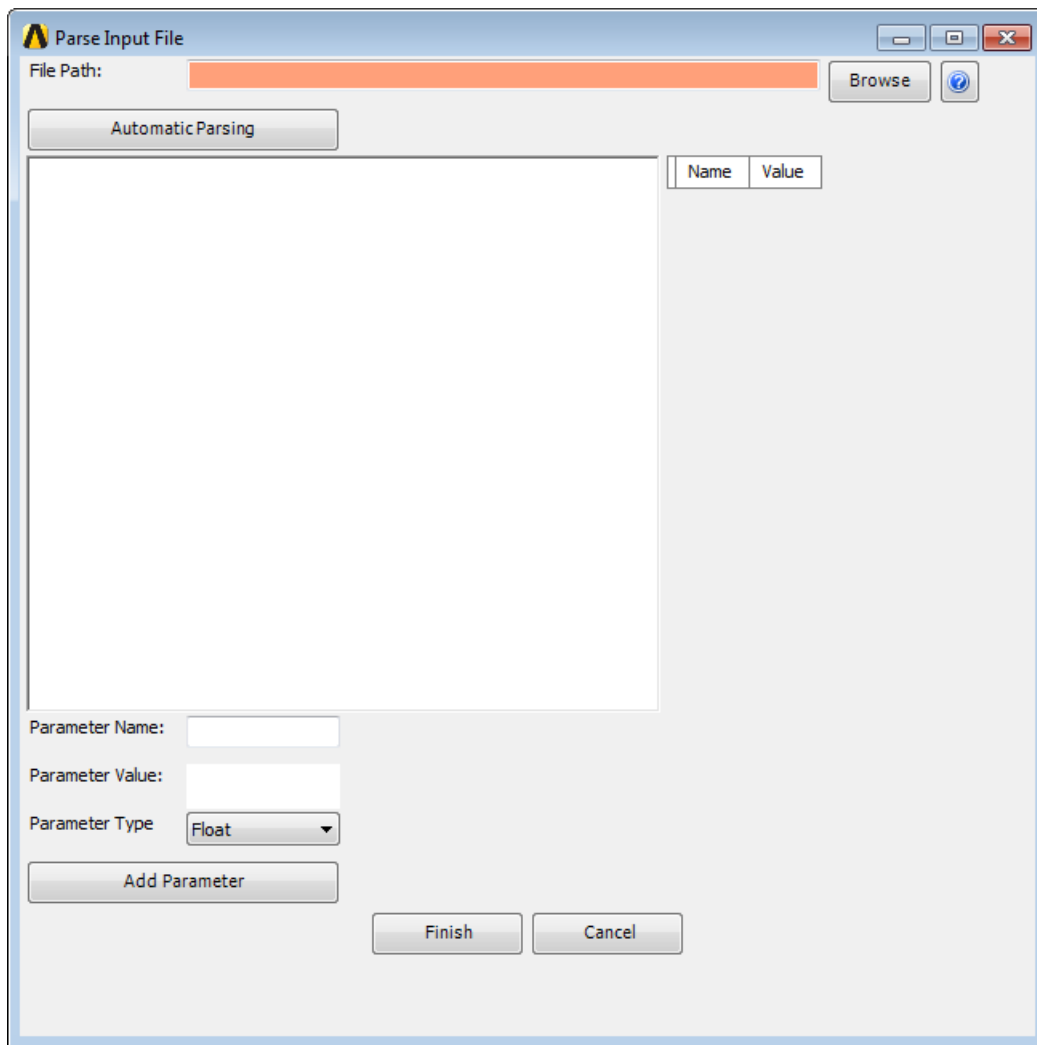
2 MyCustomTask

MyCustomTaskGroup

Properties of Schematic A2: MyCustomTask

	A	B
1	Property	Value
2	General	
3	Always Include in Design Point Update	<input type="checkbox"/>
4	Component ID	MyCustomTask
5	Directory Name	MyWorkflow
6	Notes	
7	Notes	
8	Used Licenses	
9	Last Update Used Licenses	
10	Application Properties	
11	Target Executable	ExampleAddinExternalSolver.exe
12	Target Executable Directory	E:\WorkflowDesignerInputFiles
13	Command Line Arguments	
14	Input Specifications	
15	Input Files	
16	Input Parameters	
17	Output Specifications	
18	Output Files	
19	Output Parameters	

2. Specify the input file:
 - a. Select **Input Files**, which is a link for opening the **Parse Input File** dialog box. To the right of the **Browse** button is a help button . Clicking it provides general information about how to set parameters automatically or manually.



- b. For **File Path**, browse to and select the input file.

The **Parse Input File** dialog box displays the content of the file, which should include a list of values that can be used as parameters.

3. To parse the entire input file and set input parameters automatically, click **Automatic Parsing**. The table on the left side of the dialog box updates to display the input parameters.

Otherwise, do the following for each value that you want to manually set as an input parameter:

- a. With the mouse cursor, click the value that you want to include in the **Parameter Set** bar.

Parameter Value displays the selected value, which will be editable in the **Parameter Set** bar.

- b. Complete **Parameter Name** and **Parameter Type**, specifying the name and data type to use for this input parameter.
 - c. Click **Add Parameter**. The table on the left side of the dialog box updates to include the selected input parameter.
4. When finished adding input parameters, click **Finish**.

In the **Project Schematic**, a **Parameters** cell is added to the custom workflow and the **Parameter Set** bar is also added. An upward arrow connects the **Parameter Set** bar to the **Parameters** cell to indicate that input parameters are defined.

Workbench converts the input file to a template, adding a suffix of - **Template** to the end of the filename. This template is necessary as a baseline because filenames change during design point updates.

Workbench copies this template to the project's local directory. To see all files in this directory, you can select **View** → **Files** to add the **Files** pane to Workbench. In the **Files** pane, you can then right-click in the **Location** column for the template file and select **Open Containing** to open the directory in which the template file resides.

5. In the **Properties** pane for the task cell, specify the output file in the same manner as you did the input file in step 2, only select **Output Files** in the **Properties** pane to open the **Parse Output File** dialog box.
6. Take the same actions indicated in step 3 to either automatically or manually select output parameters.
7. When finished adding output parameters, click **Finish**.

In the **Project Schematic**, a downward arrow now connects the **Parameters** cell to the **Parameter Set** bar to indicate that output parameters are defined.

Workbench converts the output file to a template, adding a suffix of - **Template** to the end of the filename. Workbench copies this template to the project's local directory, which you can see by performing the steps described in step 4.

8. In the **Properties** pane for the task cell, specify command line arguments for the external application:
 - a. Select **Command Line Arguments**, which is a link for opening a dialog box.
 - b. In the entry field, specify the arguments needed by the external application.

This might be as simple as clicking **Input File** and **Output File** to create the following argument:
[Input File][Output File]

However, you can also add a switch such as **-b** for batch execution or a flag for using a particular feature.

- c. When finished, click **Submit**.

After performing these steps, no properties for the task cell should be highlighted, which means that you should be able to update the task cell successfully.

Because Workbench uses templates for the input and output files, you can add design points in the **Table** pane of the **Parameter Set** bar and update all design points to generate results for output parameters. Then, if you have a license for ANSYS DesignXplorer and would like to further explore a particular design point, from the Workbench **Toolbox** under **Design Exploration**, you can insert any DesignXplorer system.

Creating an Example Custom Workflow for a “Squares” Application

This topic shows how to use the **ACT Workflow Designer** to create an integrated “Squares” application. This external application accepts an input text file and output text file as command line arguments. The input file contains a double precision number, preceded by the string `input=`. On reading and parsing an input value, the application computes the square of this value. The application then writes this result to the output file specified as a command line argument. In the output file, the string `output=` precedes the result value.

You can create this same custom workflow by downloading the compressed file [CustomSquaresApp.zip](#) and then extracting the files that it contains to a directory that you can access.

Create and Publish the Custom Workflow

1. Start Workbench.
2. In the **Toolbox** under **ACT**, double-click **Create a workflow**. The **ACT Workflow Designer** starts, opening on a new Workbench tab.
3. In the **Toolbox** pane, double-click **Add a task group**. A task group cell is added to the **Workflow Preview** workspace.
4. Select this cell and then specify task group properties in the **Properties** pane:
 - a. For **Name**, type `Squares`.
 - b. For **Version**, leave the default value, which is `1`.
 - c. For **Icon**, browse to the directory to which you extracted files, select `squares.png`, and click **Open**.
 - d. For **Category**, leave the default value, which is **ACT**.
 - e. For **Abbreviation**, type `Squares`.
 - f. For **Parametric**, leave the check box cleared.
5. In the **Toolbox** pane, double-click **Add a task** to add a task cell below the task group cell.
6. Select this cell and then specify task properties in the **Properties** pane:
 - a. For **Name**, type `Run Solver`.
 - b. For **Version**, leave the default value, which is `1`.
 - c. For **Icon**, browse to the directory to which you extracted files, select `squares_component.png`, and click **Open**.
 - d. For **Application Path**, browse to the directory to which you extracted files, select `ExampleAd-dinExternalSolver.exe`, and click **Open**.

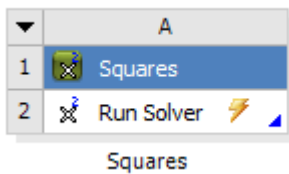
Green check marks appear to the right of both cells, indicating that the minimum properties required for publishing are defined.

7. Select **Workflow Actions Publish Workflow**. The **Publishing Settings** dialog box opens.
8. For **Workflow Name**, type **Squares**.
9. Click **Publish**.

Once the workflow is published, the **Workflow Designer** closes, returning you to the Workbench **Project Schematic**. In the Workbench **Toolbox** under **ACT**, the entry **Squares** is now added. Additionally, the files for the ACT extension named **Squares** are created here: `C:\Users\YourUserName\AppData\Local\Roaming\Ansys\vXXX\ACT\extensions`, where `vXXX` is the active ANSYS version.

Add the Custom Workflow to the Project Schematic and Configure

1. To add this custom workflow to the **Project Schematic**, in the Workbench **Toolbox** under **ACT**, double-click **Squares**.



2. Right-click the **Run Solver** cell and select **Properties**.

In the **Properties** pane, you will see three properties highlighted: **Command Line Arguments**, **Input Files**, and **Output Files**.

3. Specify the input file and automatically add the single input parameter that this file contains:
 - a. In the **Properties** pane, click **Input Files**, which opens the **Parse Input File** dialog box.
 - b. For **File Path**, browse to the directory to which you extracted files, select `input.txt`, and click **Open**.

The dialog box displays the single line in this file: `input=1.0`.

- c. To add this single value to the **Parameter Set** bar as an input parameter, click **Automatic Parsing**.

The table in the upper right of the dialog box displays this name and value.

	Name	Value
X	input_1	1.0

- d. Click **Finish**. In the Workbench **Project Schematic**, a **Parameters** cell is added to the custom workflow and the **Parameter Set** bar is shown. An upward arrow connects the **Parameter Set** bar to the **Parameters** cell.
4. Specify the output file and automatically add the single output parameter that this file contains:
 - a. In the **Properties** pane, click **Output Files**, which opens the **Parse Output File** dialog box.

- b. For **File Path**, browse to the directory to which you extracted files, select `output.txt`, and click **Open**.

The dialog box displays the single line in this file: `output=0.0`.

- c. To add this single value to the **Parameter Set** bar as an output parameter, click **Automatic Parsing**.

The table in the upper right of the dialog box displays this name and value.

	Name	Value
X	output_1	0.0

- d. Click **Finish**. In the Workbench **Project Schematic**, a downward arrow now connects the **Parameters** cell to the **Parameter Set** bar.

5. Specify command line arguments:

- a. In the **Properties** pane, click **Command Line Arguments**, which opens a dialog box for entering arguments. For this example, you need to specify only references to the input and output files.
- b. Click **Input File** and then click **Output File**.

The entry field now contains `[Input File][Output File]`

- c. Click **Submit**.

No properties are now highlighted. With your custom workflow configured, you are ready to insert the data to process as design points.

Insert Additional Design Points

In the **Parameter Set** bar, you specify the data that the external application is to process as design points.

1. In the Workbench **Project Schematic**, double-click the **Parameter Set** bar to open it.

In the **Table** pane, you see the one input value that was in the input file:

Table of Design Points						
	A	B	C	D	E	F
1	Name ▼	P1 - Run Solver::Input ▼	P2 - Run Solver::Output ▼	Retain	Retained Data	Note ▼
2	DP 0 (Current)	1	⚡	<input checked="" type="checkbox"/>	✓	
*				<input type="checkbox"/>		

2. Insert three more design points with input values of 2, 3, and 4 as shown:

Table of Design Points						
	A	B	C	D	E	F
1	Name ▾	P1 - Run Solver::Input ▾	P2 - Run Solver::Output ▾	<input type="checkbox"/> Retain	Retained Data	Note ▾
2	DP 0 (Current)	1	⚡	<input checked="" type="checkbox"/>	✓	
3	DP 1	2	⚡	<input type="checkbox"/>		
4	DP 2	3	⚡	<input type="checkbox"/>		
5	DP 3	4	⚡	<input type="checkbox"/>		
*				<input type="checkbox"/>		

- In the toolbar, click **Update All Design Points**. The executable specified for this custom workflow runs calculating an output for each design point.

When the update finishes, the **Table** pane displays a result for each design point. In this example, the result is the square of the input value.

Table of Design Points						
	A	B	C	D	E	F
1	Name ▾	P1 - Run Solver::Input ▾	P2 - Run Solver::Output ▾	<input type="checkbox"/> Retain	Retained Data	Note ▾
2	DP 0 (Current)	1	1	<input checked="" type="checkbox"/>	✓	
3	DP 1	2	4	<input type="checkbox"/>		
4	DP 2	3	9	<input type="checkbox"/>		
5	DP 3	4	16	<input type="checkbox"/>		
*				<input type="checkbox"/>		

- Close the **Parameter Set** bar.
- Close Workbench, saving the project only if you want to refer to it later.

Unless you delete the extension **Squares**, an entry for this custom workflow remains in the Workbench **Toolbar** under **ACT**. This means that you can always add this custom workflow to the **Project Schematic** and then configure and use it as indicated in this example.

Subsequent sections provide technical information for creating much more complex custom workflows and describes the manual creation process.

Global Callbacks

In ACT workflow extensions, callbacks invoke custom actions within the Workbench project workflow. Global callbacks are available for both **Project Schematic**-level actions and Workbench-level actions. While **Project Schematic**-level actions are executed in the **Project Schematic**, Workbench-level actions are executed elsewhere in the project, such as in the **Parameter Set** bar.

The following topics describe how to invoke custom actions and use global callbacks:

[Invoking Custom Project Schematic Actions](#)

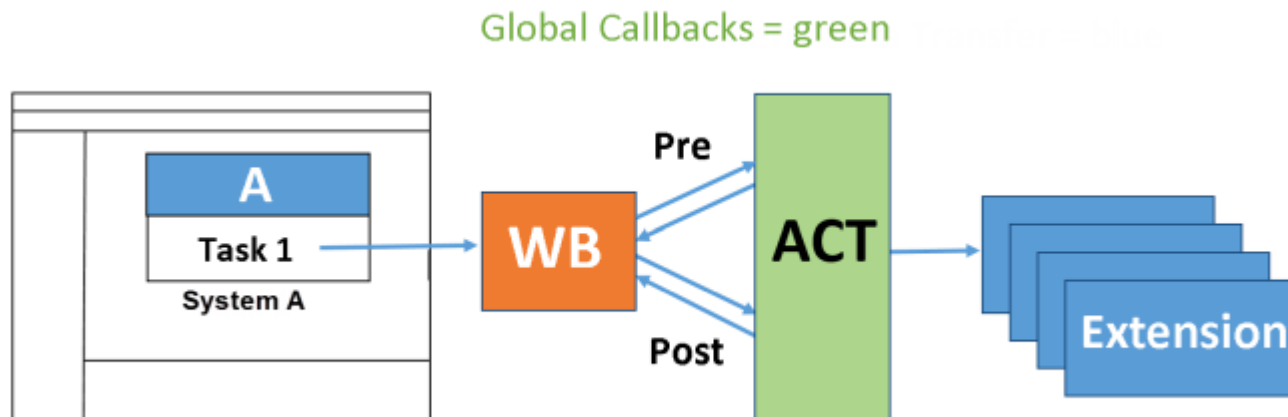
[Invoking Custom Workbench Actions](#)

[Manipulating the Observation Process of Global Callbacks](#)

Invoking Custom Project Schematic Actions

ACT global callbacks for **Project Schematic** actions provide hooks that listen and respond to activity in the Workbench **Project Schematic**. You can use these hooks to define custom processes or actions to be executed at specific times in the Workbench **Project Schematic** workflow.

Callback support for **Project Schematic** actions is provided by the **Action Observation** feature. This feature provides notification before and after every action processed by the **Project Schematic**. This enables ACT to identify a point in the workflow so that you can specify that a custom action be executed either before or after that point. Both pre-action and post-action callback support has been added to eleven distinct **Project Schematic** actions (such as **Create**, **Update**, **Refresh**, and so on.)



ACT does not observe **Project Schematic** activities until at least one extension provides one callback for a **Project Schematic** action. At that time, ACT starts observing the requested **Project Schematic** action and invokes your callback. Most callbacks for **Project Schematic** actions receive a user task as an argument. For more information, see the [ANSYS ACT API Reference Guide](#).

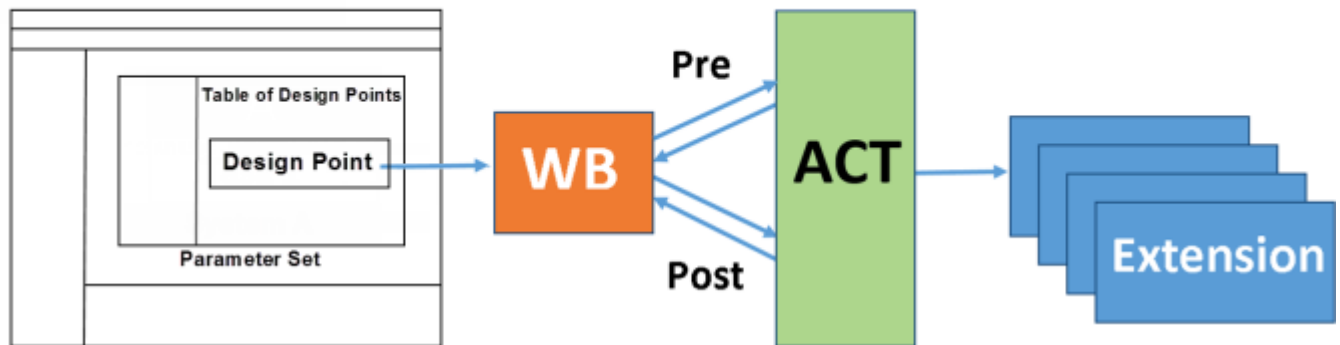
Once observations of **Project Schematic** actions have begun, a user-registered **Project Schematic** callback is invoked for every component that exists in the **Project Schematic**. This applies to both ANSYS-installed and ACT-defined tasks. For example, if you create an analysis system and update the project, ACT invokes the update-related callbacks up to n times, where n is the number of components (tasks) in the system (task group).

Invoking Custom Workbench Actions

ACT global callbacks for Workbench actions provide hooks that listen and respond to activity that occurs within the project, but outside the **Project Schematic**. You can use these hooks to define custom processes or actions to be executed at specific times in the Workbench project workflow.

Currently, pre- and post-execution callbacks are available for changes to project design points, which is executed in the table of design points in the **Parameter Set** bar.

Global Workbench Callbacks = Green



Manipulating the Observation Process of Global Callbacks

For both **Project Schematic** and Workbench actions, you can manipulate the observation process of global callbacks.

Accessing Information

For task-related callbacks, from the task, you can access name information, template information, properties, and the task's underlying container via the property **InternalObject**.

Querying

For queries, you can alter the action's return value in the post-execution callback.

Aborting an Action

To abort an action, you can throw an exception from the pre-execution callback.

Filtering Components

To process only the callback for a specific task or template, you must filter the components by performing a check inside the callback implementation based on the supplied arguments.

For example, for **Project Schematic** actions, you can access the task names or containers to decide whether or not to take action. If the action must apply only to all **Fluent Setup** components, you would create an update callback that contains the following code:

```
def myPostUpdateCallback(task):
    container = task.InternalObject
    if container != None and container.Type.Equals('Ansys.Fluent.Addin:SetupContainer'):
        #perform your post-update action on the fluent setup container
    Stopping Observations:
```

When you unload all extensions that register global callbacks, ACT stops observing the specified actions. This sets Workbench back to its prior state and can be used as a fix if either **Project Schematic** or Workbench observations cause unexpected behavior.

Data Transfers

Data transfer is the process of a pushing and pulling data between upstream and downstream tasks within the Workbench **Project Schematic**. This exchange of information allows you to fully engage simulation workflow management, project state tracking, and iterative design.

The top-down, left-right information exchange within a workflow can appear as both *implicit* (between tasks stacked on top of each other in a task group) or *explicit* (between tasks from separate task groups).

Discrete input and output data types declare which tasks can exchange data and the role they play (producer, consumer, or both).

ACT supports three data type categories to engage data transfer:

ANSYS-installed data types

Use the data types exposed by ANSYS products to exchange information between your task and ANSYS products.

Custom input and output types

Define custom data types to exchange information between two or more ACT custom tasks.

Generic transfer

Leverage simplified data type-less transfer between ACT custom tasks and as input to **Model**, **Setup**, and **Solution** cells in Mechanical-based **Analysis Systems**.

For information on using task properties to define data transfers within the **Project Schematic**, see [Defining Task-Level Data Transfer](#) (p. 63).

For extension examples that demonstrate data transfer methods, see the following topics:

- [Generic Mesh Transfer](#) (p. 99)
- [Custom Transfer](#) (p. 101)
- [Generic Material Transfer](#) (p. 103)

For information that you need to transfer data between ANSYS-installed data types, see the following appendices:

- [Appendix A: Component Input and Output Tables](#) (p. 111)
- [Appendix B: ANSYS-Installed System Component Template and Display Names](#) (p. 187)
- [Appendix C: Data Transfer Types](#) (p. 209)
- [Appendix D: Addin Data Types and Data Transfer Formats](#) (p. 215)
- [Appendix E: ANSYS-Installed Custom Workflow Support](#) (p. 219)

Progress Monitoring

ACT provides the property **ExtAPI.UserInterface.ProgressMonitor** for monitoring the progress of custom workflows. This property returns an object of type **IProgressMonitor**.

The following APIs are available:

Class	Member	Description
IProgressMonitor	WorkName	The name of the current work under progress monitoring.

Class	Member	Description
	WorkStatus	The current status for the work under progress monitoring.
	WorkDetails	The current details for the work under progress monitoring.
	TotalWorkUnits	The total number of work units allocated to the work under progress monitoring.
	CompletedWorkUnits	The completed number of work units.
	Parent	The parent progress monitor, if this is a child monitor. Otherwise, null.
	PropagateWorkStatus	Indicates whether to propagate a child's work status up to the parent.
	PropagateWorkDetails	Indicates whether to propagate a child's work details up to the parent.
	CanAbort	Indicates whether the operation can be cancelled.
	CanInterrupt	Indicates whether the current operation can be interrupted.
	BeginWork(name, units)	Starts work engaged with progress monitoring.
	UpdateWork(units)	Updates the current work units.
	EndWork()	Marks the current work as complete (progress at 100%).
	CreateChildMonitor(units)	Creates a child progress monitor with the current monitor as its parent.
	Status	<p>The current status of this progress monitor.</p> <p>ProgressStatus enum values:</p> <ul style="list-style-type: none"> • NotStarted • Running • Aborted • Interrupted • Completed

A sample implementation of progress monitoring functionality follows.

```
def update(task):
    monitor = ExtAPI.UserInterface.ProgressMonitor
    monitor.BeginWork("Testing 123...", 3)
    LogProgress(monitor)
    monitor.WorkStatus = "Test Status!"
    monitor.WorkDetails = "Test Details!"
    System.Threading.Thread.Sleep(2000)
    monitor.UpdateWork(1)
    LogProgress(monitor)
    monitor.WorkDetails = "More details..."
    System.Threading.Thread.Sleep(2000)
    monitor.UpdateWork(1)
    LogProgress(monitor)
    monitor.WorkDetails = "Even more details..."
    System.Threading.Thread.Sleep(2000)
    monitor.UpdateWork(1)
    LogProgress(monitor)
    monitor.WorkDetails = "Final details..."
    System.Threading.Thread.Sleep(2000)
    monitor.EndWork()
def LogProgress(monitor):
    ExtAPI.Log.WriteMessage("Progress: " + str(monitor.CompletedWorkUnits)
                           + "/" + str(monitor.TotalWorkUnits))
```

Process Utilities

ACT process utilities enable you to execute any outside program with a single method call. To do this, you call the method **start** exposed on the object **ExtAPI.ProcessUtils**.

The following APIs are available:

Class	Member	Description
ExtAPI.ProcessUtils	Start(target, args)	Starts an application, file, or other target. Returns an integer error code from the resulting process. Waits for process to exit.
	Start(target, useShell, args)	Starts an application, file, or other target. useShell indicates whether or not to use the running OS's shell to execute the target.

A sample implementation of process utilities follows.

```
import System

def update(task):
    activeDir = task.ActiveDirectory
    extensionDir = task.Extension.InstallDir
    exeName = "ExampleAddinExternalSolver.exe"
    solverPath = System.IO.Path.Combine(extensionDir, exeName)

    inputValue = task.Properties["Inputs"].Properties["Input"].Value

    inputFileName = "input.txt"
    outputFileName = "output.txt"
    dpInputFile = System.IO.Path.Combine(activeDir, inputFileName)
```

```

dpOutputFile = System.IO.Path.Combine(activeDir, outputFileName)

#write input file
f = open(dpInputFile, "w")
f.write('input='+inputValue.ToString(System.Globalization.NumberFormatInfo.InvariantInfo))
f.close()

exitCode = ExtAPI.ProcessUtils.RunApplication(solverPath, dpInputFile, dpOutputFile)
if exitCode != 0:
    raise Exception ('External solver failed!')
#read output file

outputValue = None
f = open(dpOutputFile, "r")
currLine = f.readline()
while currLine != "":
    valuePair = currLine.split('=')
    outputValue = System.Double.Parse(valuePair[1], System.Globalization.NumberFormatInfo.InvariantInfo)
    currLine = f.readline()
f.close()

if outputValue == None:
    raise Exception("Error in update - no output value detected!")
else:
    task.Properties["Outputs"].Properties["Output"].Value = outputValue

```

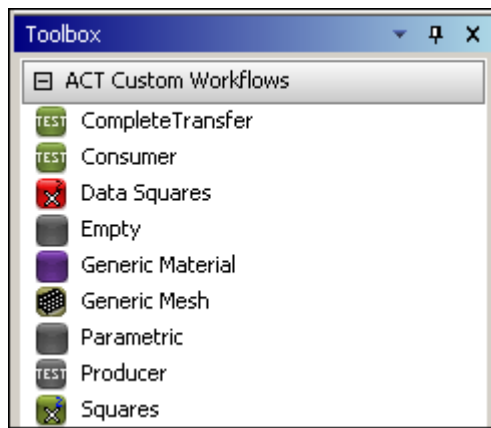
Manual Creation of a Custom Workflow

Manually creating the custom task group and tasks to expose in a Workbench workflow involves the creation, installation, and loading of an ACT workflow extension.

The steps to manually create a custom workflow extension are the same as for any other type of ACT extension. At minimum, you must create the XML definition file, IronPython script, and any support files, such as task group and task images and custom application executables. You must then place the script and support files in an extension directory at the same level as the XML file. The [ACT Developer's Guide](#) provides information on [creating extensions](#) and [setting up the directory structure](#).

Before you can use an extension you've created, you must install it and load it into Workbench via the [Extension Manager](#). Once the extension is loaded, the functionality defined in it becomes available.

- If global callbacks are defined, they are invoked in the **Project Schematic** workflow or other Workbench project location as defined. Each global callback is invoked each time the operation associated with it is performed.
- If a custom task group and tasks are defined, the custom task group is exposed as a system in the Workbench **Toolbar** under **ACT Custom Workflows**.



You can add custom task groups to the **Project Schematic**. When you invoke the **Update** operation on a custom task within the task group, the extension can:

- Obtain the inputs
- Prepare the inputs
- Write input files
- Run the external solver or perform calculations
- Read output files
- Set the parameters or properties to the calculated values

XML Extension Definition for a Custom Workflow

The extension's XML definition file specifies the custom workflow within Workbench, including task groups (systems) designed for a particular simulation objective. All analyses performed within Workbench begin by referencing a task group. The XML file describes the task groups and declares the contained tasks.

At the extension level, the XML file must specify, at minimum:

- Extension name
- Interface context (must be **Project**)
- Folder for storing images for task group or task icons
- Workflow definition

At the workflow level, the XML file can specify:

- Workflow context (required and must be **Project**)
- Tasks (container for individual task definitions)
- Task group (container for the task group definition)

- Global callbacks

At the global callback level, the XML file can specify pre- and post-operation callbacks for **Project Schematic** and Workbench operations. For more information, see [Defining a Global Callback \(p. 56\)](#) and [Global Callbacks \(p. 46\)](#).

At the task level, the XML file can specify:

- Task name (required)
- Task version (required)
- Display text
- Image name
- Context menus
- Callbacks
- Property groups and properties
- Inputs and outputs
- Parameters
- Remote job execution specifications

For more information, see [Defining a Task \(p. 61\)](#).

At the task group level, the XML file can specify:

- Task group name (required)
- Task group version (required)
- Header text
- Toolbox category
- Parametric support
- Image name
- Abbreviation

For more information, see [Defining a Task Group \(p. 81\)](#).

Basic Structure of the XML Definition File

The following example of an XML file shows the basic structure:

```
<extension version="1" name="">
  <guid shortid="">69d0095b-e138-4841-a13a-de12238c83f3</guid>
  <script src="" />
  <interface context="Project">
```



```

    <images>images</images>
  </interface>
  <workflow name="" context="Project" version="1">
<callbacks>
  <onbeforetaskreset></onbeforetaskreset>
  <onaftertaskreset></onaftertaskreset>
  <onbeforetaskrefresh></onbeforetaskrefresh>
  <onaftertaskrefresh></onaftertaskrefresh>
  <onbeforetaskupdate></onbeforetaskupdate>
  <onaftertaskupdate></onaftertaskupdate>
  <onbeforetaskduplicate></onbeforetaskduplicate>
  <onaftertaskduplicate></onaftertaskduplicate>
  <onbeforetasksourceschanged></onbeforetasksourceschanged>
  <onaftertasksourceschanged></onaftertasksourceschanged>
  <onbeforetaskcreation></onbeforetaskcreation>
  <onaftertaskcreation></onaftertaskcreation>
  <onbeforetaskdeletion></onbeforetaskdeletion>
  <onaftertaskdeletion></onaftertaskdeletion>
  <onbeforetaskcanusetransfer></onbeforetaskcanusetransfer>
  <onaftertaskcanusetransfer></onaftertaskcanusetransfer>
  <onbeforetaskcanduplicate></onbeforetaskcanduplicate>
  <onaftertaskcanduplicate></onaftertaskcanduplicate>
  <onbeforetaskstatus></onbeforetaskstatus>
  <onaftertaskstatus></onaftertaskstatus>
  <onbeforetaskpropertyretrieval></onbeforetaskpropertyretrieval>
  <onaftertaskpropertyretrieval></onaftertaskpropertyretrieval>
  <onbeforedesignpointchanged></onbeforedesignpointchanged>
  <onafterdesignpointchanged></onafterdesignpointchanged>
</callbacks>
<tasks>
  <task name="" caption="" icon="" version="1">
    <callbacks>
      <onupdate></onupdate>
      <onrefresh></onrefresh>
      <oninitialize></oninitialize>
      <onedit></onedit>
      <onreset></onreset>
      <onstatus></onstatus>
      <onreport></onreport>
    </callbacks>
    <contextmenus>
      <entry name="" caption="" icon="" priority="" type="">
        <callbacks>
          <onclick></onclick>
        </callbacks>
      </entry>
    </contextmenus>
    <propertygroup name="" caption="">
      <property name="" caption="" control="" default="" readonly="" needupdate="" visible="" persis
    </propertygroup>
    <property name="" caption="" control="" default="" readonly="" needupdate="" visible="" persistent
    <parameters>
      <parameter name="" caption="" usage="" control="" version="1"/>
    </parameters>
    <inputs>
      <input/>
      <input type="" format=""/>
    </inputs>
    <outputs>
      <output type="" format=""/>
    </outputs>
    <rsmjob name="" deletefiles="" version="1">
      <inputfile id="1" name=""/>
      <outputfile id="1" name=""/>
      <program>
        <platform name="" path=""/>
        <argument name="" value="" separator=""/>
      </program>
      <callbacks>
        <oncreatejobinput></oncreatejobinput>
        <onjobstatus></onjobstatus>

```

```

        <onjobcancellation></onjobcancellation>
        <onjobreconnect></onjobreconnect>
    </callbacks>
</rsmjob>
</task>
</tasks>
<taskgroups>
    <taskgroup name="" caption="" icon="" category="" abbreviation="" version="1" isparametricgroup="False">
        <includetask name="" external=""/>
        <includeGroup name=""/>
    </taskgroup>
</taskgroups>
</workflow>
</extension>

```

Defining a Global Callback

In the element **<workflow>**, one or more global callbacks (either for **Project Schematic** or Workbench operations) can be defined in a child element **<callbacks>**.

ACT currently provides 11 **Project Schematic** operations, each of which supports both a pre-operation and post-operation callback. ACT also provides one non-**Project Schematic** Workbench operation for changing design points, which also supports both a pre-operation and post-operation callback.

The basic structure of a global callback definition follows.

```

<callbacks>
  <onbeforetaskreset></onbeforetaskreset>
  <onaftertaskreset></onaftertaskreset>
  <onbeforetaskrefresh></onbeforetaskrefresh>
  <onaftertaskrefresh></onaftertaskrefresh>
  <onbeforetaskupdate></onbeforetaskupdate>
  <onaftertaskupdate></onaftertaskupdate>
  <onbeforetaskduplicate></onbeforetaskduplicate>
  <onaftertaskduplicate></onaftertaskduplicate>
  <onbeforetasksourceschanged></onbeforetasksourceschanged>
  <onaftertasksourceschanged></onaftertasksourceschanged>
  <onbeforetaskcreation></onbeforetaskcreation>
  <onaftertaskcreation></onaftertaskcreation>
  <onbeforetaskdeletion></onbeforetaskdeletion>
  <onaftertaskdeletion></onaftertaskdeletion>
  <onbeforetaskcanusetransfer></onbeforetaskcanusetransfer>
  <onaftertaskcanusetransfer></onaftertaskcanusetransfer>
  <onbeforetaskcanduplicate></onbeforetaskcanduplicate>
  <onaftertaskcanduplicate></onaftertaskcanduplicate>
  <onbeforetaskstatus></onbeforetaskstatus>
  <onaftertaskstatus></onaftertaskstatus>
  <onbeforetaskpropertyretrieval></onbeforetaskpropertyretrieval>
  <onaftertaskpropertyretrieval></onaftertaskpropertyretrieval>
  <onbeforedesignpointchanged></onbeforedesignpointchanged>
  <onafterdesignpointchanged></onafterdesignpointchanged>
</callbacks>

```

See the following topics for additional information:

[Invoking Global Callbacks](#)

[Filtering Global Callbacks](#)

[Specifying Global Callback Execution Order](#)

[Available Project Schematic Callbacks](#)

[Available Workbench Callbacks](#)

Invoking Global Callbacks

By default, if a global **Project Schematic** callback is defined in the XML file, it is called for each instance of the **Project Schematic** task associated with it. This applies to both ANSYS-installed and ACT-defined tasks. For example, if you have defined the callback `<onaftertaskupdate>`, it is invoked for each task that is updated. If three tasks in the **Project Schematic** are updated, the callback is invoked three times, once after each update.

Unless otherwise noted, a global **Project Schematic** workflow callback receives a task as the method argument. Depending on the callback, the task is the concrete task displayed in the **Project Schematic** or the task template.

Filtering Global Callbacks

To specify that a callback is processed only for a specific location in the project, such as the task or template, you can perform a check within the callback implementation based on the supplied arguments. For example, in the following IronPython code sample:

- The method `onBeforeUpdate` is unfiltered. The actions are invoked before the update of each eligible task in the **Project Schematic**.
- The method `onAfterUpdate` is filtered. It produces different messages for **Engineering Data** tasks and the second **Setup** task, while printing 'ignored' for all other tasks.

Note:

The **Setup** tasks for multiple Fluent task groups would be identified in the following sequence: **Setup**, **Setup 1**, **Setup 2**, and so on.

- The method `onBeforeDuplicate` is filtered. It prints a specific message for the **Setup** tasks for all Fluent tasks on the **Project Schematic**.

```
...
def onBeforeUpdate(task):
    msg = getPrintMessage('pre-update', task)
    print msg

def onAfterUpdate(task):
    if task.Name == "Engineering Data":
        msg = getPrintMessage('post-update', task)
        print msg
    elif task.Name == "Setup 2":
        print("fluent setup 2")
    else:
        print("ignored")

def onBeforeDuplicate(task):
    if task.Container.Type == "Ansys.Fluent.Addin:SetupContainer":
        print("all fluent setups")
    else:
        print("ignored")
...
```

Specifying Global Callback Execution Order

When multiple callbacks are defined for the same pre- or post-operation **Project Schematic** action, you can specify their order of execution by adding the **order** attribute to the callback declaration:

```
<workflow ...>
  <callbacks>
    <onbeforetaskreset order="1">...</onbeforetaskreset>
    ...
  </callbacks>
</workflow>
```

ACT executes the callbacks from lowest order to highest order. This means that a callback with an order of 2 executes after a callback with an of order of 1. When values are identical, the order in which the extension is loaded dictates the collision resolution.

Available Project Schematic Callbacks

This section describes the global callback and corresponding methods and arguments for each **Project Schematic** operation.

Reset

Resets a component back to its pristine, new state.

Global Callbacks	Methods	Arguments
<onbeforetaskreset>	onBeforeReset	task
<onaftertaskreset>	onAfterReset	

Refresh

Consumes all upstream data and prepares any local data for an ensuing update.

Global Callbacks	Methods	Arguments
<onbeforetaskrefresh>	onBeforeRefresh	task
<onaftertaskrefresh>	onAfterRefresh	

Update

Generates all broadcast output types that render the component fully solved.

Global Callbacks	Methods	Arguments
<onbeforetaskupdate>	onBeforeUpdate	task
<onaftertaskupdate>	onAfterUpdate	

Create

Creates a task based on an underlying template.

Global Callbacks	Methods	Arguments
<onbeforetaskcreate>	onBeforeCreate	templateTask Note: This is a task representing the

Global Callbacks	Methods	Arguments
		template. Some components do not include the template as a parameter to the command create . As a result, this could be null in some situations.
<onaftertaskcreate>	onAfterCreate	task

Delete

Removes a task from a task group.

Global Callbacks	Methods	Arguments
<onbeforetaskdelete>	onBeforeDelete	task
<onaftertaskdelete>	onAfterDelete	

Duplicate

Creates an identical, yet independent, clone of the task.

Global Callbacks	Methods	Arguments
<onbeforetaskduplicate>	onBeforeDuplicate	task
<onaftertaskduplicate>	onAfterDuplicate	

SourcesChanged

Processes a change in upstream sources.

Global Callbacks	Methods	Arguments
<onbeforetasksourceschanged>	onBeforeSourcesChanged	task
<onaftertasksourceschanged>	onAfterSourcesChanged	

CanUseTransfer

Checks whether the task can consume data from a specific upstream task.

Callback receives two tasks as arguments: the *source* (upstream producer) task and *target* (downstream consumer) task.

Global Callbacks	Methods	Arguments
<onbeforetaskcanusetransfer>	onBeforeCanUseTransfer	sourceTask, targetTask
<onaftertaskcanusetransfer>	onAfterCanUseTransfer	sourceTask, targetTask, systemCalculatedCanUse Note: The argument systemCalculatedCanUse provides the result that the original task determined. The return value of this

Global Callbacks	Methods	Arguments
		callback (Boolean) alters the originally calculated value.

CanDuplicate

Checks whether the task permits duplication.

Callback receives no arguments. The **Project Schematic** does not provide any workable information to construct or provide tasks (either container-based or template-based). Callback only serves as a general “processing” hook if exposed.

Global Callbacks	Methods	Arguments
<onbeforetaskcanduplicate>	onBeforeCanDuplicate	None
<onaftertaskcanduplicate>	onAfterCanDuplicate	systemCalculatedCanUse Note: The argument systemCalculatedCanDuplicate provides the result that the original task determined. The return value of this callback (Boolean) alters the originally calculated value.

Status

Calculates the task’s current state.

Global Callbacks	Methods	Arguments
<onbeforetaskstatus>	onBeforeStatus	task
<onaftertaskstatus>	onAfterStatus	task, systemCalculatedStatus Note: The argument systemCalculatedStatus provides the result that the original task determined. The return value of this callback component state alters the originally calculated value.

PropertyRetrieval

Determines the visibility of property-containing objects.

Global Callbacks	Methods	Arguments
<onbeforetaskpropertyretrieval>	onBeforePropertyRetrieval	task

Global Callbacks	Methods	Arguments
<code><onaftertaskpropertyretrieval></code>	<code>onAfterPropertyRetrieval</code>	task, systemCalculatedPropertyObject Note: The argument <code>systemCalculatedPropertyObjects</code> provides the result that the original task determined. The return value of the callback (list of data references) alters the originally calculated value.

Available Workbench Callbacks

This section describes the global callback and corresponding methods for the non-**Project Schematic** Workbench operation for changing design points:

`designpointchanged`

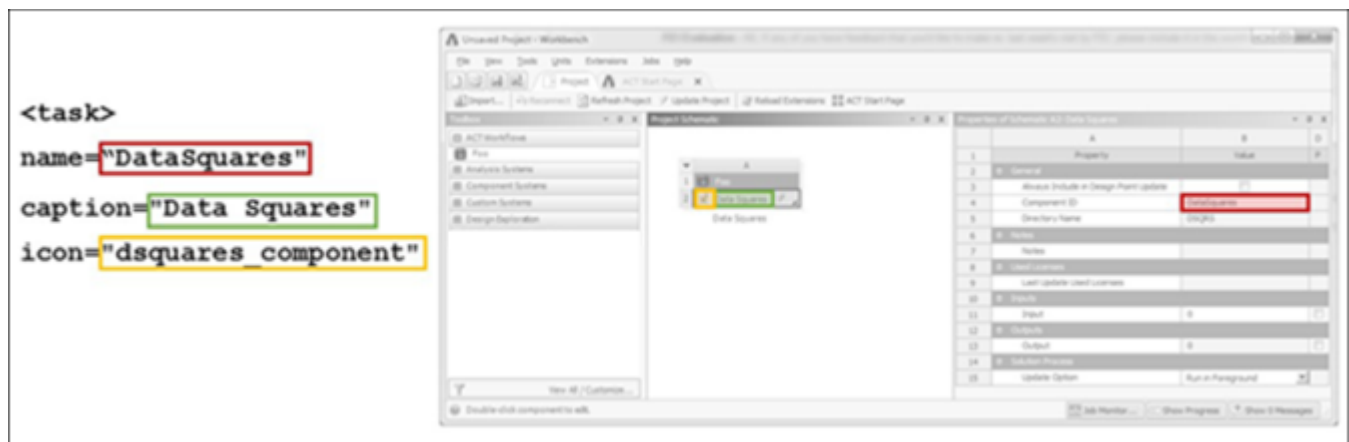
Signals to workflow that a design point change is about to or has occurred.

Global Callbacks	Methods
<code><onbeforedesignpointchanged></code>	<code>onBeforeDesignPointChanged</code>
<code><onafterdesignpointchanged></code>	<code>onAfterDesignPointChanged</code>

Defining a Task

One or more custom tasks can be defined in the element `<tasks>`. Each task is defined in a child element `<task>`. The base `<task>` class represents an extension-based workflow task that serves as a template from which ACT creates real instances.

In each task definition, you must include the attributes **name** and **version**. The following figure maps the task **DataSquares** to the Workbench interface.



Child elements are used to specify callbacks, inputs and outputs, properties, parameters, context menus, and remote execution. The basic structure of a task definition follows:

```
<tasks>
  <task name="" caption="" icon="" version="1" enableGenericTransfer="">
    <callbacks>
      <IsVisible></IsVisible>
      <IsValid></IsValid>
      <GetValue></GetValue>
      <SetValue></SetValue>
      <OnShow></OnShow>
      <OnHide></OnHide>
      <OnInit></OnInit>
      <OnAdd></OnAdd>
      <OnRemove></OnRemove>
      <OnDuplicate></OnDuplicate>
      <OnActivate></OnActivate>
      <OnCancel></OnCancel>
      <OnApply></OnApply>
      <Value2String></Value2String>
      <String2Value></String2Value>
      <OnValidate></OnValidate>
      <OnMigrate></OnMigrate>
    </callbacks>
    <contextmenus>
      <entry name="" caption="" icon="" priority="" type="">
        <callbacks>
          <onclick></onclick>
        </callbacks>
      </entry>
    </contextmenus>
    <propertygroup name="" caption="">
      <property name="" caption="" control="" default="" options="" readonly="" needupdate="" visible="" persistent="">
    </propertygroup>
    <property name="" caption="" control="" default="" readonly="" needupdate="" visible="" persistent="" isparameter="">
      <callbacks>
        <IsVisible></IsVisible>
        <IsValid></IsValid>
        <GetValue></GetValue>
        <SetValue></SetValue>
        <OnShow></OnShow>
        <OnHide></OnHide>
        <OnInit></OnInit>
        <OnAdd></OnAdd>
        <OnRemove></OnRemove>
        <OnDuplicate></OnDuplicate>
        <OnActivate></OnActivate>
        <OnCancel></OnCancel>
        <OnApply></OnApply>
        <Value2String></Value2String>
        <String2Value></String2Value>
        <OnValidate></OnValidate>
        <OnMigrate></OnMigrate>
      </callbacks>
    </property>
    <parameters>
      <parameter name="" caption="" usage="" control="" version="1"/>
    </parameters>
    <inputs>
      <input/>
      <input type="" count="" format=""/>
    </inputs>
    <outputs>
      <output type="" format=""/>
    </outputs>
    <rsmjob name="" deletefiles="" version="1">
      <inputfile id="1" name=""/>
      <outputfile id="1" name=""/>
    </rsmjob>
    <program>
      <platform name="" path=""/>
    </program>
  </task>
</tasks>
```



```

    <argument name="" value="" separator="" />
  </program>
  <callbacks>
    <oncreatejobinput></oncreatejobinput>
    <onjobstatus></onjobstatus>
    <onjobcancellation></onjobcancellation>
    <onjobreconnect></onjobreconnect>
  </callbacks>
</rsmjob>
</task>
</tasks>

```

See the following topics for additional information:

[Defining Task-Level Attributes](#)

[Defining Task-Level Data Transfer](#)

[Defining Task-Level Data Transfer Access](#)

[Managing Task-Level Files](#)

[Defining Task-Level Callbacks](#)

[Defining Task-Level Context Menus](#)

[Defining Task-Level Property Groups and Properties](#)

[Accessing and Invoking Task-Related Workbench Data](#)

Defining Task-Level Attributes

All attributes set on ACT-defined tasks are persisted and resumed within a project.

```

...
task = ExtAPI.DataModel.Tasks[0]
task.SetAttributeValue('MyAttribute', 'MyValue')
Save(FilePath='...', Overwrite=True)
Reset()
Open(FilePath='...')
task = ExtAPI.DataModel.Tasks[0]
task.GetAttributeValue('MyAttribute')

```

Defining Task-Level Data Transfer

ACT provides you with a method of transferring data within the **Project Schematic**. All tasks contain a transfer object of type **GeneralTransfer**. This object contains one property, **TransferData**, of type **Dictionary<string, object>**. When the type is fully created for each task, ACT automatically exposes the type as an output by default. If you do not want to engage in all transfer possibilities enabled through the transfer, you can declare this at the task level:

```

<workflow name="MyWorkflow" context="Project" version="1">
  <tasks>
    <task name="MyTask" ... enableGenericTransfer="False">
      ...
    </task>
    ...
  </tasks>
</workflow>

```

Once a transfer-enabled task exists in the **Project Schematic**, the transfer data can be accessed to push and pull data within the *data store*, which is the repository for storing and managing the data used by ANSYS services and products.

A simple property **TransferData** is exposed off of the task argument of all task callbacks or tasks retrieved from **ExtAPI** or as a callback argument:

```
def producer_update(task):
    data = task.TransferData
```

Once obtained, the dictionary **TransferData** acts like any other dictionary within ACT. You can both set and retrieve values through string keys, and you can perform other collection-based calls such as **Add()** and **Remove()**. The custom **IDictionary** is coded in such a way that if you try to access a non-existent key, such as when setting a new value, it automatically adds the key and allows the set to take place.

```
task.TransferData["Test"] = "Sample text"
```

On the consumer side, you can easily access the transfer data:

```
def consumer_update(task):
    container = task.InternalObject
    upstreamData = container.GetInputDataByType(InputType="GeneralTransfer")
    for upstreamDataEntity in upstreamData:
        task = ACT.GetTaskForContainer(upstreamDataEntity.Container)
        data = task.TransferData["Test"]
```

The transfer data approach eliminates dependency on file and file path transfers. However, if desired, you can still opt to pass file paths via the new transfer data. No further actions or implementations are required to fulfill the basic framework transfer connections.

Note:

Downstream processing might require additional actions implemented via ACT workflow callbacks.

In some circumstances, you might want to interrogate a task's source tasks. For example, this can ease the access of transfer data from upstream sources. The property **SourceTasks** is available on all objects of the type **UserTask**:

```
def myTaskUpdate(task):
    sources = task.SourceTasks
    for sourceTask in sources:
        data = sourceTask.TransferData["Test"]
```

Tasks can filter or block their source connections by implementing a task-level callback **canconsumedata**.

XML Definition File:

```
<task name="Consumer"...>
  <callbacks>
    <canconsumedata>taskCanUse</canconsumedata>
    ...
  </callbacks>
  ...
</task>
```

IronPython Script:

```
def taskCanUse(task, sourceTask):
    if sourceTask.Name == "Foo":
        return True
    else:
        return False
```

For a list of ANSYS-installed tasks that support general data transfer, see [Appendix E: ANSYS-Installed Custom Workflow Support \(p. 219\)](#).

Custom solver-based systems also support this type of data transfer.

Defining Task-Level Data Transfer Access

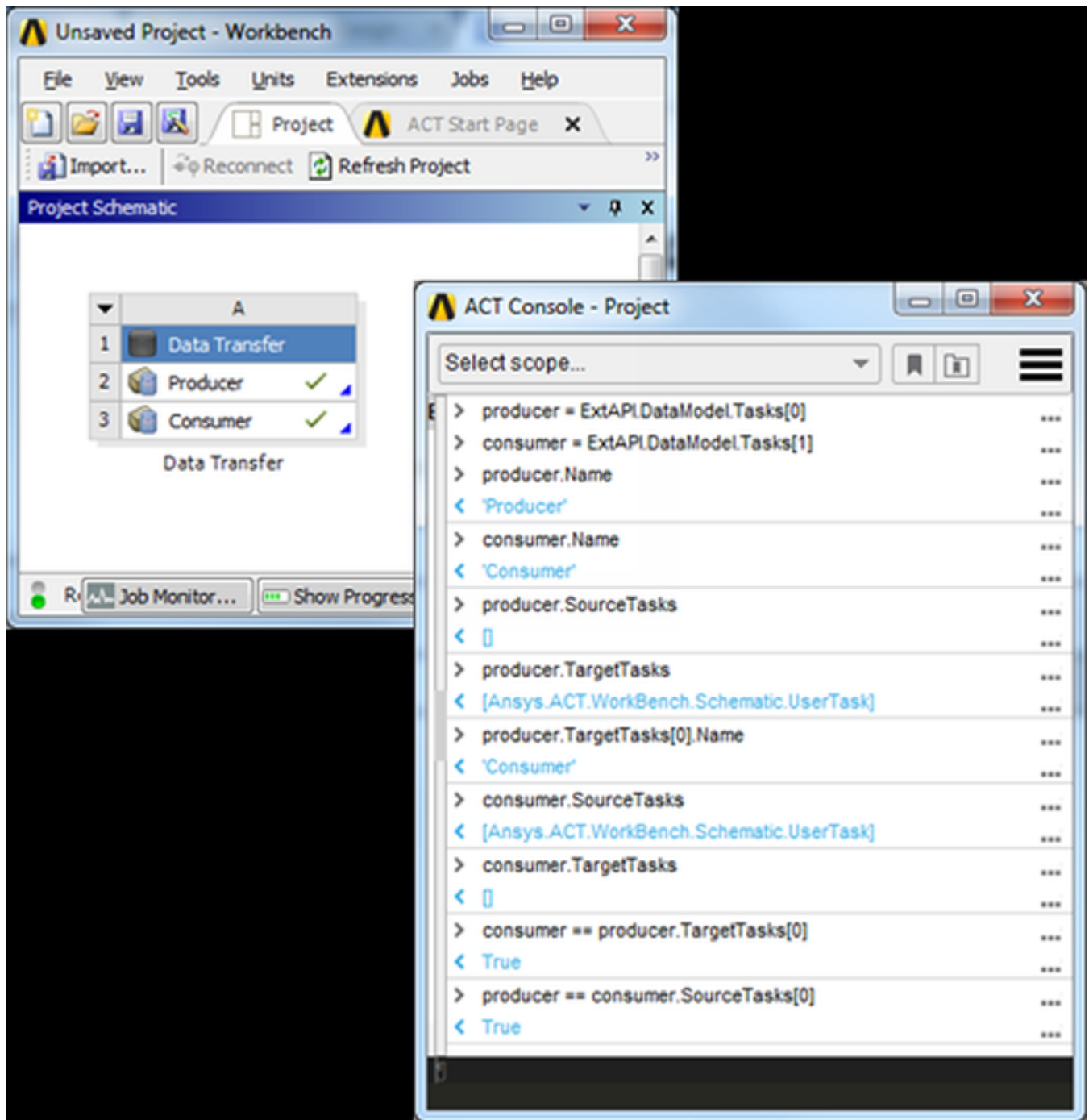
The properties **InputData** and **OutputData** are exposed on the object **UserTask**. Additionally, **UserTask** exposes a property **TargetTasks** that returns all downstream tasks that consume (hold a connection to) the current task.

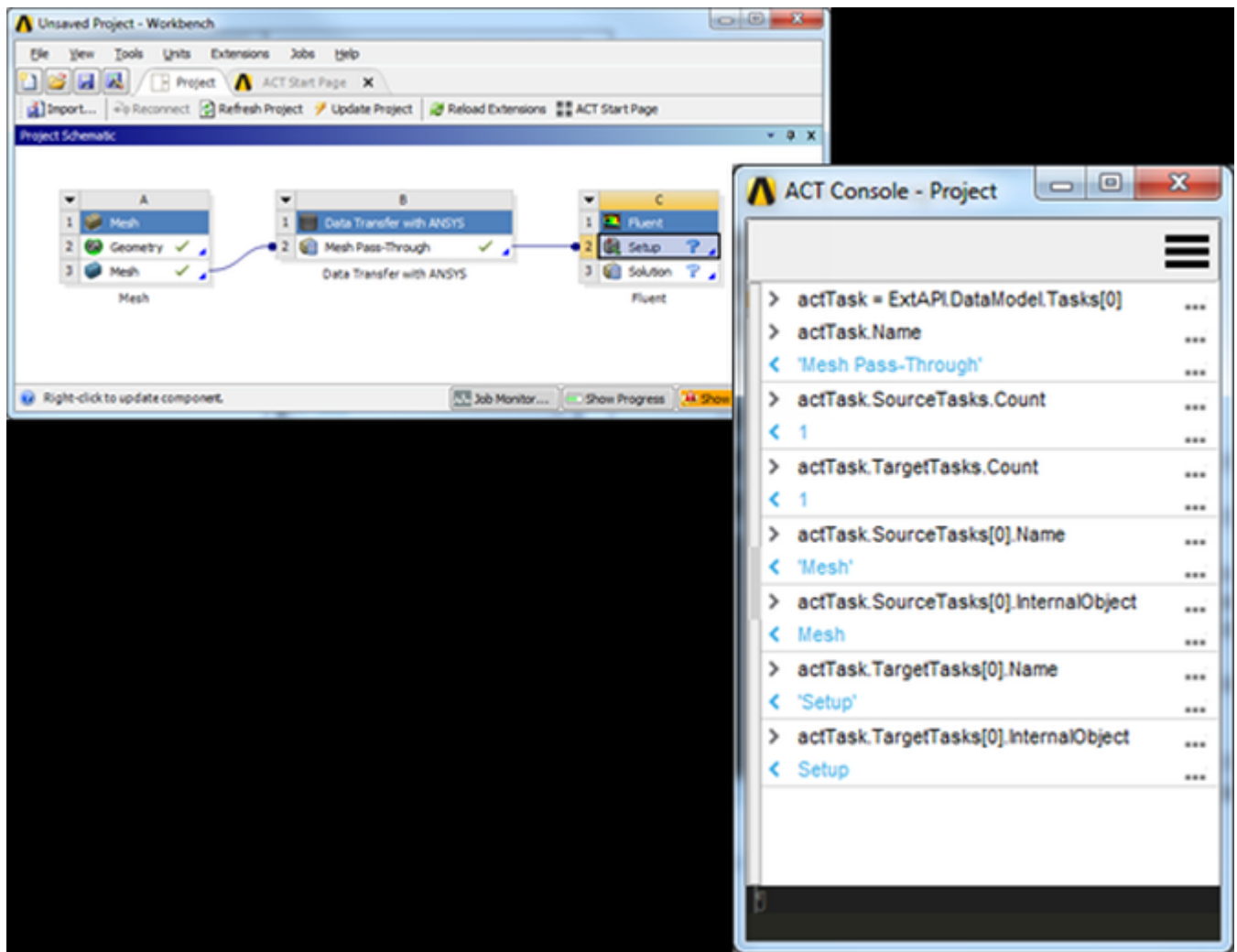
Class	Member
UserTask	InputData
	OutputData
	TargetTasks

An example follows.

```
import System

def consumer_update(task):
    upstreamData = task.InputData["MyData"]
    fileRef = None
    upstreamDataCount = upstreamData.Count
    if upstreamDataCount > 0:
        fileRef = upstreamData[0]
        task.UnregisterFile(fileRef)
        task.RegisterFile(fileRef.Location)
def producer_update(task):
    extensionDir = task.Extension.InstallDir
    activeDir = task.ActiveDirectory
    filePath = System.IO.Path.Combine(extensionDir, "Sample_Materials.xml")
    activeFilePath = System.IO.Path.Combine(activeDir, "Sample_Materials.xml")
    task.UnregisterFile(activeFilePath)
    if System.IO.File.Exists(activeFilePath) == False:
        System.IO.File.Copy(filePath, activeFilePath)
    fileRef = task.RegisterFile(activeFilePath)
    myData = task.OutputData["MyData"][0]
    myData.TransferFile = fileRef
def mesh_update(task):
    upstreamData = task.InputData["MeshingMesh"]
    meshFileRef = None
    upstreamDataCount = upstreamData.Count
    if upstreamDataCount > 0:
        meshFileRef = upstreamData[0]
    meshOutput = task.OutputData["SimulationGeneratedMesh"][0]
    meshOutput.TransferFile = meshFileRef
```





Managing Task-Level Files

The class **UserTask** includes APIs to facilitate easier management of task-level files. The following APIs are available:

Class	Member	Description
UserTask	ActiveDirectory	Obtains the active design point directory for the task.
	RegisteredFiles	A collection of all registered files associated with the task.
	RegisterFile(filepath)	Registers and associates a file with the task.
	UnregisterFile(filePath)	Unregisters and disassociates a file from the task.
	UnregisterFile(filePath, delete)	Unregisters and disassociates a file from the task.
	UnregisterFile(fileReference)	Unregisters and disassociates a file from the task.

Class	Member	Description
	UnregisterFile(fileReference, delete)	Unregisters and disassociates a file from the task.

A sample implementation of task-level file management capabilities follows.

```
import System

def update(task):
    activeDir = task.ActiveDirectory
    installDir = task.Extension.InstallDir
    srcImagePath = System.IO.Path.Combine(installDir,
                                           System.IO.Path.Combine("images", "logo-ansys.jpg"))
    destImagePath = System.IO.Path.Combine(activeDir, "logo-ansys.jpg")
    task.UnregisterFile(destImagePath)
    System.IO.File.Copy(srcImagePath, destImagePath)
    task.RegisterFile(destImagePath)
    ExtAPI.Log.WriteMessage('task "' + task.Name + '" Registered File Count = '
                           + str(task.RegisteredFiles.Count))
```

Note:

The **UnRegister(...)** members fully unregister the files only if they have been associated with the task (through the **task.Register(...)** members). Otherwise, the files remain registered.

Defining Task-Level Callbacks

Callbacks to IronPython functions are specified in the element **<callbacks>**. All callbacks receive, at minimum, a task object as an argument. The following table lists the available callbacks and their arguments.

Callback	Arguments
oninitialize	task
onupdate	task
onrefresh	task
onreset	task
onedit*	task
onstatus**	task
onreport***	task, report
ondelete	task
canconsumedata	sourceTask, targetTask

*Definition of the callback **<onedit>** automatically creates a default **Edit** context menu for the task.

The callback **<onstatus> invokes the method **status** when the application asks the task for its current state. For more information, see [Accessing State-Handling APIs \(p. 14\)](#).

***The callback `<onreport>` invokes the method `report` when you generate a project report. It allows the task to access the report object and add its own task-specific reporting content. For more information, see [Accessing Project Reporting APIs \(p. 16\)](#).

In the event of task-level callback errors, exceptions are provided to the Workbench framework, allowing the framework to detect failures and correctly manage task states. Exception details are also reported to the log.

Defining Task-Level Context Menus

If you specify the callback `<onedit>` in the XML file, a default **Edit** context menu option is automatically created for the task. However, it is possible to define additional interface operations for each task.

Custom interface operations are specified in the optional element `<contextmenus>`. At minimum, the context menu entry definition must include the attribute `name`. When the optional attribute `type` is defined, it must be set to `ContextMenuEntry`.

Each entry in the element `<contextmenus>` must have a callback `<onclick>`, which receives a task object as its argument.

The basic structure of the element `<contextmenus>` follows.

```
<contextmenus>
  <entry name="" caption="" icon="" priority="" type="">
    <callbacks>
      <onclick></onclick>
    </callbacks>
  </entry>
</contextmenus>
```

Defining Task-Level Property Groups and Properties

Instead of using parameters to drive your simulation, you can work through the data model, simplifying data access by defining custom properties.

Properties can be defined in the optional element `<propertygroup>`. At minimum, each property group definition must include the attribute `name`. The attribute `caption` is used as the group category name in the Workbench **Property** pane if the child properties are visible.

A property is defined in the element `<property>`, which can be either a child to the element `<propertygroup>` or a stand-alone definition at the same level. At minimum, each property definition must include the attributes `name` and `control`.

The basic structure of a property definition follows.

```
<propertygroup name="" caption="">
  <property name="" caption="" control="" default="" readonly="" needupdate="" visible="" persistent="" isparam="">
</propertygroup>
<property name="" caption="" control="" default="" readonly="" needupdate="" visible="" persistent="" isparam="">
```

ACT supports the following property control types:

- **string/text**

- `double`
- `float`
- `integer`
- `DataReference`
- `boolean`
- `object`
- `quantity`
- `option/select`
- `fileopen`
- `folderopen`
- `list`
- `dictionary`
- `DataContainerReference`

Defining Task-Level Property Callbacks

Callbacks to IronPython functions can be specified in a task-level property definition. All property callbacks receive, at minimum, the entity (task) and property as arguments. Available property callbacks are listed and described.

IsVisible

You can define the callback `isvisible` on task-exposed properties. This callback can dynamically control whether or not to display a property. You can key off of any desired condition or other task property to determine the property's visibility, returning `True` or `False` as a result.

XML Definition File:

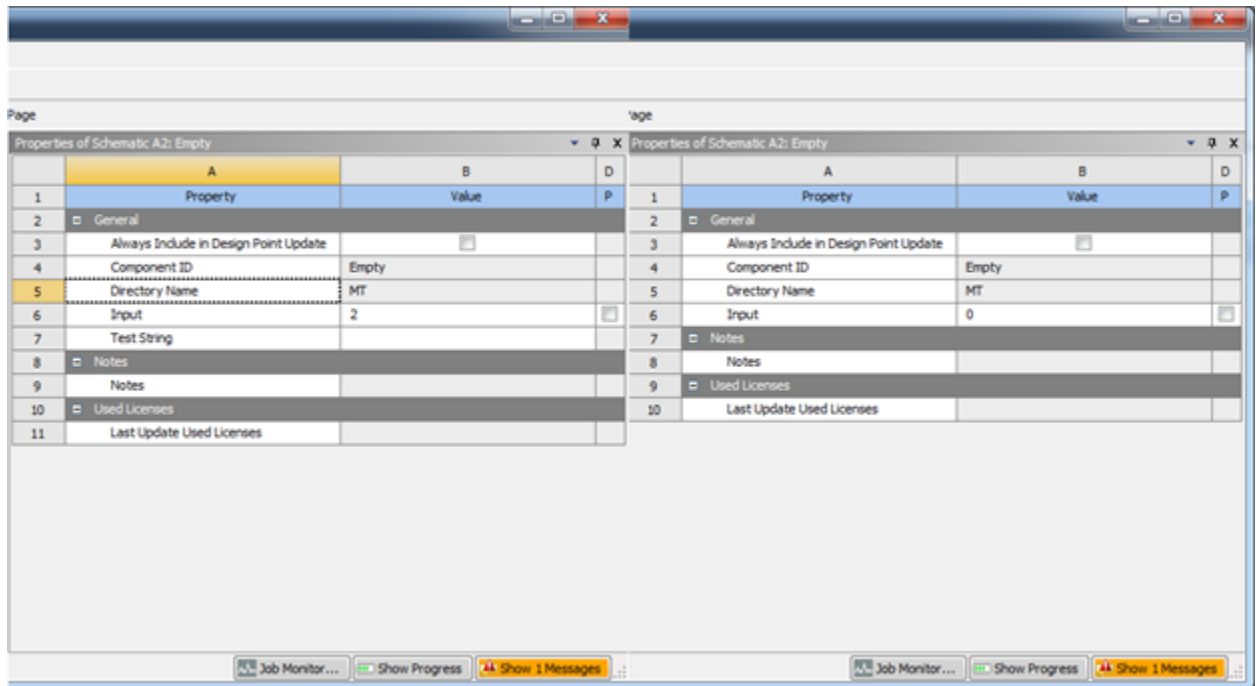
```
...
<task ...>
  ...
  <property name="Input" caption="Input" control="double" .../>
  <property name="HiddenString" caption="Test String" control="string" ...>
    <callbacks>
      <isvisible>stringVisibilityCheck</isvisible>
    </callbacks>
  </property>
  ...
</task>
...
```

IronPython Script:

```
def stringVisibilityCheck(entity, property):
    inputVal = entity.Properties["Input"]
    if inputVal.Value == 2.0:
        return True
```



```
else:
    return False
```



IsValid

You can define the callback **isvalid** on task-exposed properties. This callback can dynamically control whether or not a property is valid. You can key off of any desired condition or other task property to determine the property's validity, returning **True** or **False** as a result.

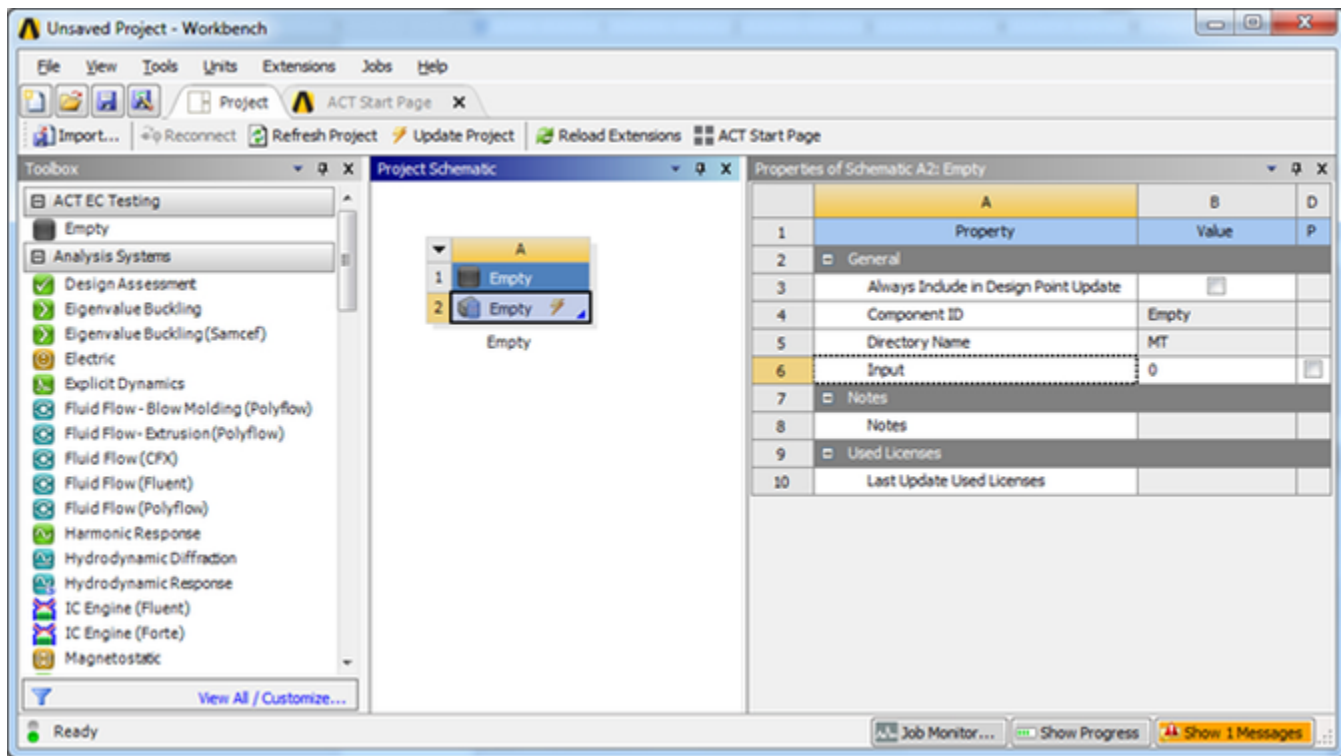
XML Definition File:

```
...
<task ...>
  ...
  <property name="Input" caption="Input" control="double" default="0.0" readonly="False" needupdate="T
    <callbacks>
      <isvalid>validityCheck</isvalid>
    </callbacks>
  </property>
  ...
</task>
...
```

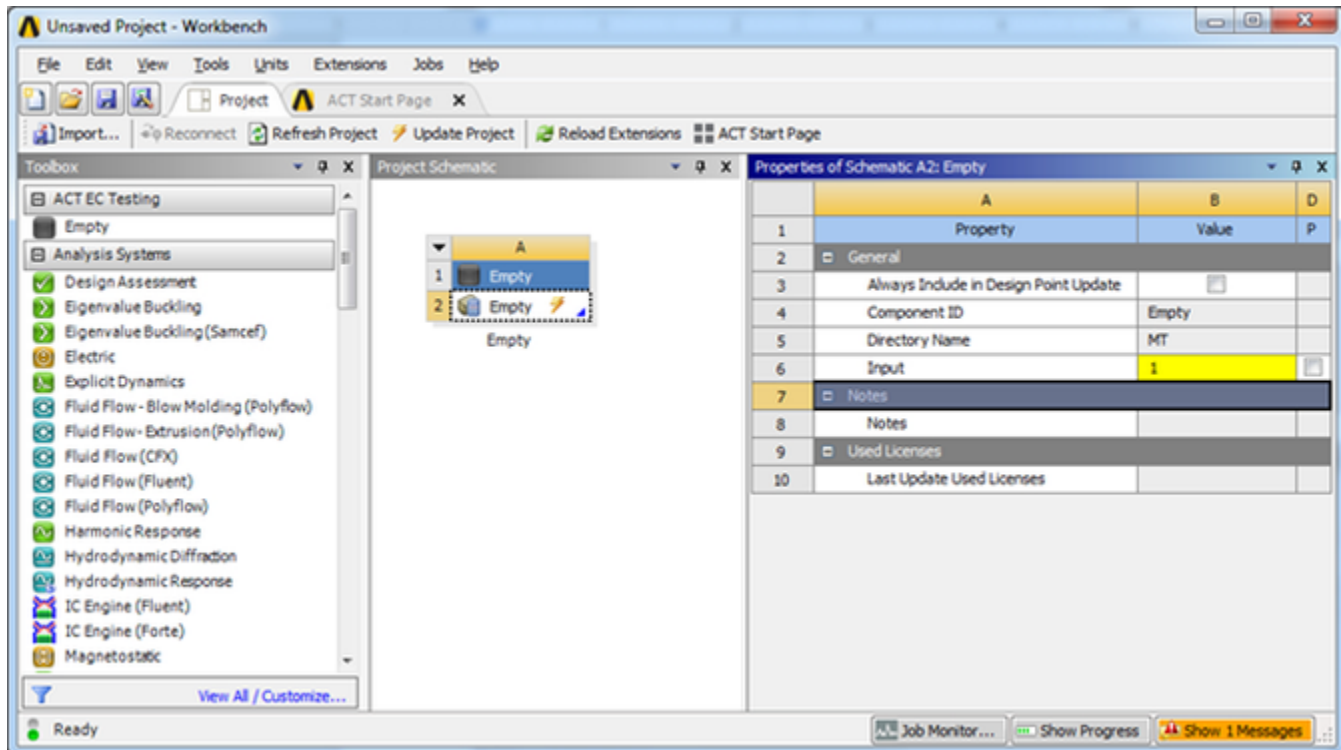
IronPython Script:

```
def validityCheck(entity, property):
    if property.Value == 1.0:
        return False
    else:
        return True
```

In the following figure, the property is valid.



In the following figure, the property is invalid.



GetValue

You can define the callback `getValue`, which is invoked when a value is obtained from the command line, a script, or through the project itself.

SetValue

You can define the callback **setvalue**, which is called when a value is set from the command line, a script, or through the project itself.

OnShow

You can define the callback **onshow**, which is called when the **Project** page is about to display the property.

OnHide

You can define the callback **onhide**, which is called when the **Project** page is about to hide the property.

OnInit

You can define the callback **oninit**, which is called when the property is first created on the task object.

****This is called at the same location as the callback **onadd**.****

OnAdd

You can define the callback **onadd**, which is called when the property is first added on the task object.

****This is called at the same location as the callback **oninit**.****

OnRemove

You can define the callback **onremove**, which is called when the property is deleted from the task object.

OnDuplicate

You can define the callback **onduplicate**, which is called when the property is duplicated to another task.

OnActivate

You can define the callback **onactivate**, which is called when the property is activated in the user interface.

****This is called at the same location as the callback **onshow**.****

OnCancel

You can define the callback **oncancel**, which is called when a property is deactivated.

****This is called at the same location as the callback **onhide**.****

OnApply

You can define the callback **onapply**, which is called immediately after a property value change but before it is actually processed.

Value2String

You can define the callback **value2string**, which is called when the property **ValueString** is retrieved.

String2Value

You can define the callback **string2value**, which is called when the property **ValueString** is set.

OnValidate

You can define the callback **onValidate**, which is called during a property value change.

OnMigrate

As of 17.1, this callback is not supported.

Using Standardized Property Interaction

You can get and set property values using standard ACT syntax, as defined through **simEntity**. At the task-level callbacks, the provided task argument provides the entry point:

```
propA = task.Properties["PropertyA"].Value
task.Properties["PropertyA"].Value = "Foo"
```

Access is available at the master **ExtAPI** level. Assuming that only one task resides in the **Project Schematic**:

```
task = ExtAPI.DataModel.Tasks[0]
propA = task.Properties["PropertyA"].Value
task.Properties["PropertyA"].Value = "Foo"
```

Property groups are treated the same as elsewhere in ACT:

```
task = ExtAPI.DataModel.Tasks[0]
group1 = task.Properties["GroupName"]
subProp1 = group1.Properties["SubPropertyA"]
subProp1Val = subProp1.Value
```

When you must perform “advanced” **Project Schematic** interactions, you can access **task.InternalObject** to receive the task’s underlying **DataContainerReference**. This might be needed to execute Workbench-specific scripting commands. For more information, see the [Discovery AIM and Workbench Scripting Guide](#).

```
task = ExtAPI.DataModel.Tasks[0]
container = task.InternalObject
container.SendCommand(Language="Python", Command="print `test`")
```

Interactions with ANSYS-installed tasks operate in the same way. For example, if you create a Fluent analysis system on the **Project Schematic**, you could set the property **RunParallel** as follows:

```
task = ExtAPI.DataModel.Task[0]
task.Properties["RunParallel"].Value = True
```

All "set"-based actions performed via **Property.Value** calls are recorded in the journal. Using the last example with a Fluent system, the resulting journal would contain the following code:

```
# encoding: utf-8
# Release 17.1
SetScriptVersion(Version="17.1.21")
Reset()
template1 = GetTemplate(TemplateName="FLUENT")
```

```

system1 = template1.CreateSystem()
setup1 = system1.GetFluentLauncherSettings()
fluentLauncherSettings1.RunParallel = True

```

Using Path-Based Property Control Support

You can specify the controls **fileopen** and **folderopen** on task properties. On task creation and property display, a folder icon appears in the property value cell. Selecting this icon displays a dialog box for selecting a file or folder. File path and folder path access remains the same as if the property was string-based.

XML Definition File:

```

<task ...>
  ...
  <property name="Prop1" caption="File Path" control="fileopen" .../>
  <property name="Prop2" caption="Directory Path" control="folderopen" .../>
  ...
</task>

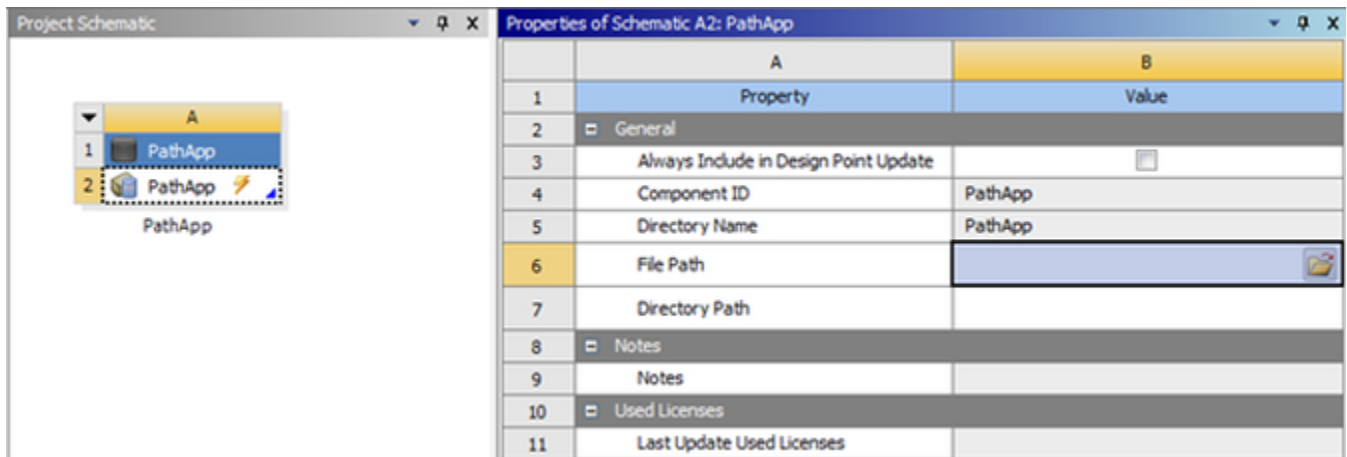
```

IronPython Script:

```

def update(task):
    filePath = task.Properties["Prop1"].Value
    dirPath = task.Properties["Prop2"].Value

```



Using String-Based and Option-Based Task Properties

For string-based task properties, specify **text** as the **control** type.

```

...
<task ...>
  ...
  <property name="StringTest" caption="String" control="text"...>
  </property>
  ...
</task>
...

```

For option-based task properties, specify the standard **select** control type and use the **options** attribute to provide the items to be populated to the control.

```

...
<task ...>
  ...
  <property name="ComboBoxTest" caption="My Prop" control="select"...>
  ...
  <attributes options="one,two,three" .../>
  ...
</task>
...

```

Specifying Property Default Values

In the XML file, you can specify default property values to be used on the **Project Schematic**.

```

...
<task ...>
  ...
  <property name="MyProperty" caption="My Property" control="text" default="Foo".../>
  ...
</task>
...

```

Defining Task-Level Parameters

You can integrate an external application into the **Project Schematic** by defining parameters that are created dynamically at initialization.

Parameters are defined in the optional element **<parameters>**. At minimum, each parameter definition must include the following attributes: **name**, **usage**, **control**, and **version**.

The basic structure of a parameter definition follows.

```

<parameters>
  <parameter name="" caption="" usage="" control="" version="1"/>
</parameters>

```

Defining Task-Level Inputs and Outputs

In a Workbench workflow, the **Project Schematic** connections serve as visual representations of data flow between tasks. These connections depend on input and output coordination. Workbench can establish connections only if an upstream (providing) task exposes outputs whose types also match the inputs for a downstream (consuming) task.

Inputs and outputs are defined in elements **<input>** and **<output>**.

You can use the attribute **type** to specify the data type expected by the connection.

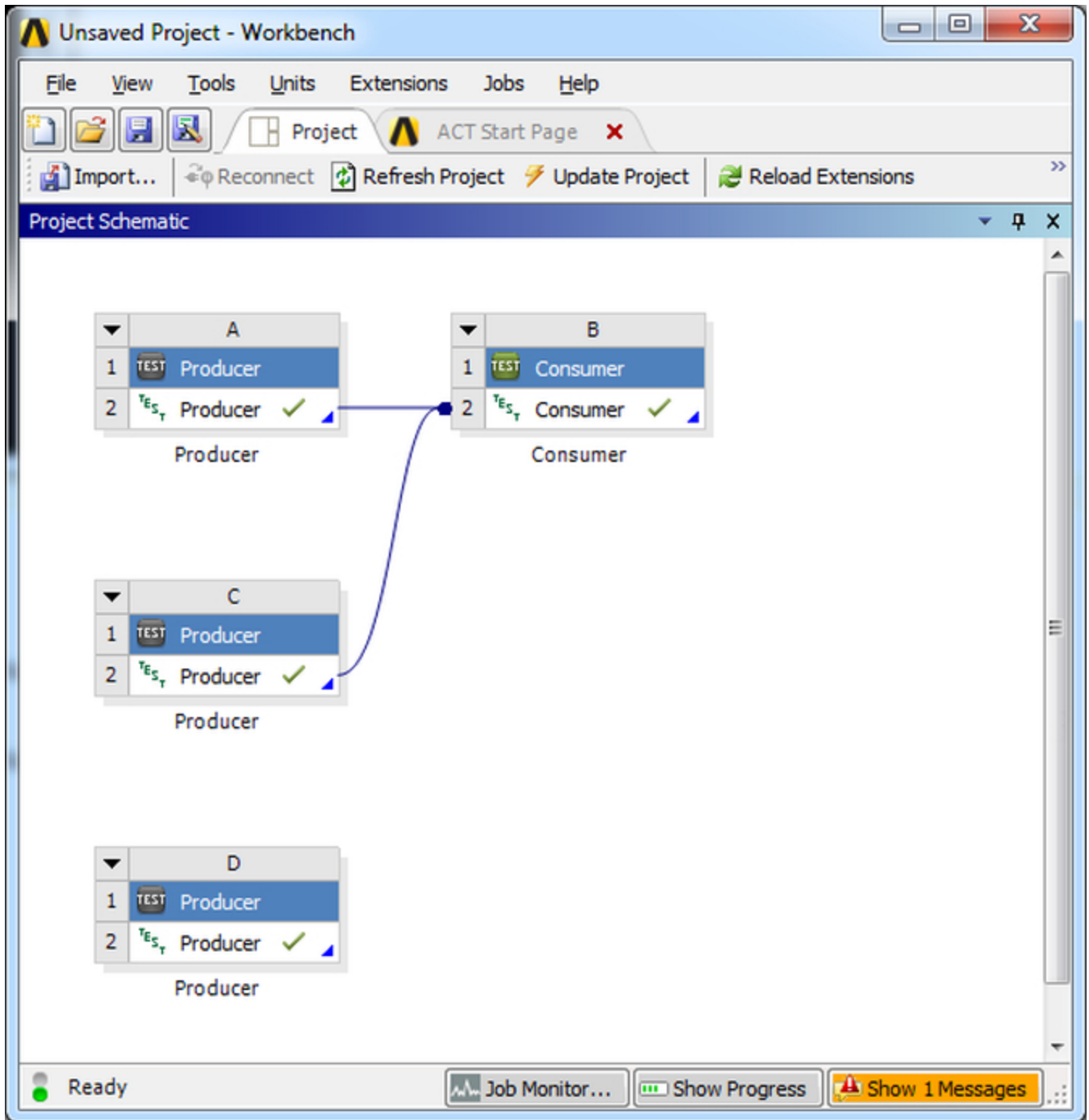
By default, ACT applies an unlimited count to the number of input specifications. You can specify a count according to your own requirements:

```

...
<task ...>
  ...
  <inputs>
    <input/>
    <input format="" type="GeneralTransfer" count=2/>
  </inputs>
  ...

```

```
</task>
...
```



Certain Workbench types require the use of both the attributes **type** and **format**. For example, a **Mesh** task that consumes a mesh and then passes a mesh to another task group would use these attributes to specify the mesh type and file format of the input and output files.

This example defines the inputs and outputs for a task within a Fluent meshing workflow. The meshing-based type values and **FluentMesh** format values instruct an upstream mesh task to output the Fluent mesh file format (MSH).

```

...
<task name="Mesher" caption="Mesher" icon="GenericMesh_cell" version="1">
  <callbacks>
    <onupdate>update</onupdate>
    <onedit>edit</onedit>
  </callbacks>
  <inputs>
    <input format="FluentMesh" type="MeshingMesh" count="1"/>
  </inputs>
  <outputs>
    <output format="FluentMesh" type="SimulationGeneratedMesh"/>
  </outputs>
</task>
...

```

For a list of supported transfer types and their corresponding transfer properties, see [Appendix C: Data Transfer Types](#) (p. 209).

For a list of the data types and data transfer formats for each addin, see [Appendix D: Addin Data Types and Data Transfer Formats](#) (p. 215).

Defining a Remote Job Execution Specification

ANSYS Remote Solve Manager (RSM) enables the remote execution of applications and processes. By leveraging high-performance clusters and efficient job scheduling, users submit jobs either as background solve tasks (local machine) or as true remote cluster and scheduler submissions. Workbench orchestrates the packaging, submission, tracking, and reintegration of job artifacts.

ACT enables you to leverage RSM capabilities, specifying the remote execution of a task's processes via the child element `<rsmJob>`. This element allows you to specify all relevant RSM-related information, including supported platforms, input and output files, process arguments, and the callbacks needed to execute the remote jobs.

This example shows the basic definition of a remote execution specification. It defines the name and version of the job, specifies that job files should be deleted after execution, and defines job input and output files and callbacks.

```

...
<task name="MyTask" caption="My Task" icon="my_task" version="1">
  ...
  <rsmjob name="squares" deletefiles="True" version="1">
    <inputfile id="1" name="input1.txt"/>
    <outputfile id="1" name="output1.txt"/>
    <program>
      <platform name="Win64" path="%AWP_ROOT171%\path_to.exe"/>
      <platform name="Linux64" path="%AWP_ROOT171%\path_to.exe"/>
      <argument name="" value="inputFile:1" separator="/">
      <argument name="" value="outputFile:1" separator="/">
    </program>
    <callbacks>
      <oncreatejobinput>createJobInput</oncreatejobinput>
      <onjobstatus>getJobStatus</onjobstatus>
      <onjobcancellation>cancelJob</onjobcancellation>
      <onjobreconnect>reconnectJob</onjobreconnect>
    </callbacks>
  </rsmjob>
</task>
...

```


For more information about how to access RSM capabilities, see [External Application Integration with Custom Data and Remote Job Execution](#) (p. 92).

Accessing and Invoking Task-Related Workbench Data

From a task, you can access and manipulate objects above the task level, such as container-level commands, Workbench-level components or parameters, and the task-containing task group:

[Executing Container-Level Commands on a Task](#)

[Accessing Workbench Parameters from a Task](#)

[Accessing an ANSYS-Installed Component from a Task](#)

[Accessing the Owing Task Group from a Task](#)

Executing Container-Level Commands on a Task

ACT provides the ability to invoke journaling and scripting-defined, container-level commands on a task. The following API is available:

Class	Member	Description
UserTask	ExecuteCommand (command-Name , args)	Invokes a Project Schematic method defined on the task.

The following code sample shows an implementation of the method **ExecuteCommand**.

```
#assume the first task is a mechanical-based system
task = ExtAPI.DataModel.Tasks[0]
task.ExecuteCommand("Edit", [])
```

Accessing Workbench Parameters from a Task

ACT enables you to access task-defined Workbench parameters directly from a task. The following API is available:

Class	Member	Description
UserTask	Parameters	Obtains the parameters defined on the task.

The following code sample shows an implementation of the method **Parameters**.

```
task = ExtAPI.DataModel.Tasks[0]
myParameters = task.Parameters
myParam = myParameters[0]
myParamValue = myParam.Value
```

Accessing an ANSYS-Installed Component from a Task

ACT enables you to access an ANSYS-installed component directly from a task. The following API is available:

Class	Member	Description
UserTask	Component	Obtains the underlying Project Schematic component associated with the task.

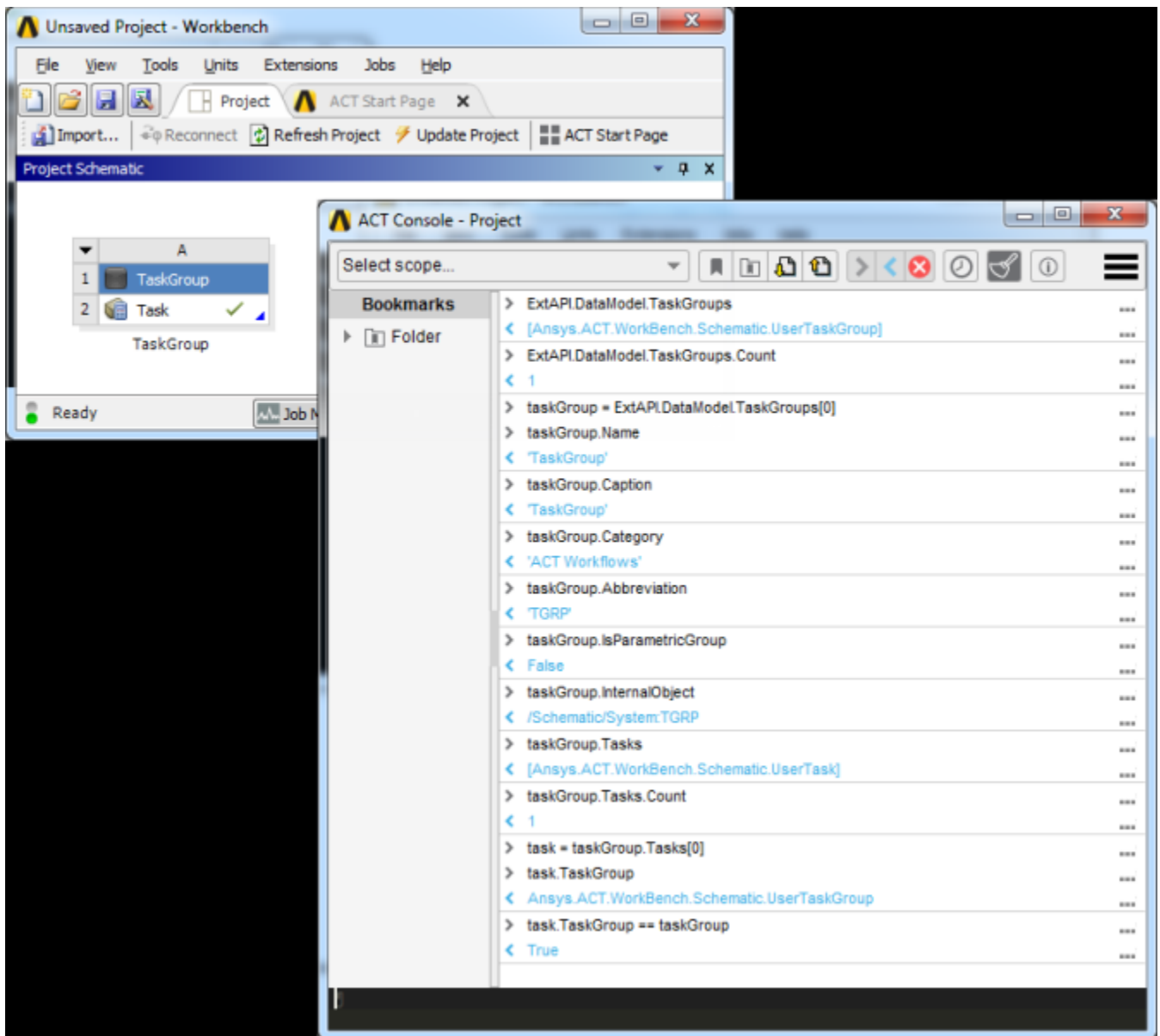
The following code sample shows an implementation of the method **Component**.

```
#assume the first task is a mechanical-based system
task = ExtAPI.DataModel.Tasks[0]
myTaskComponent = task.Component
```

Accessing the Owning Task Group from a Task

ACT enables you to access specific **TaskGroup** instances that have been created on the **Project Schematic**. Because a **task** is owned by a **TaskGroup**, you can access the owning **TaskGroup** directly from a new **Task**. The following APIs are available:

Class	Member	Description
Task (UserTask)	TaskGroup (User-TaskGroup)	Obtains the task's task group owner.
ExtAPI.DataModel	TaskGroups	The list of all task groups currently on the Project Schematic .
TaskGroup (User-TaskGroup)	Tasks (UserTasks)	The list of all tasks owned by the task group.
	Name	The task group name.
	Caption	The task group display name.
	Category	The task group toolbox category.
	Abbreviation	The task group abbreviation.
	IsParametricGroup	Indicates whether the task group falls under the Parameter Set bar, operating only on parameter and design point data.
	InternalObject	(System data reference)

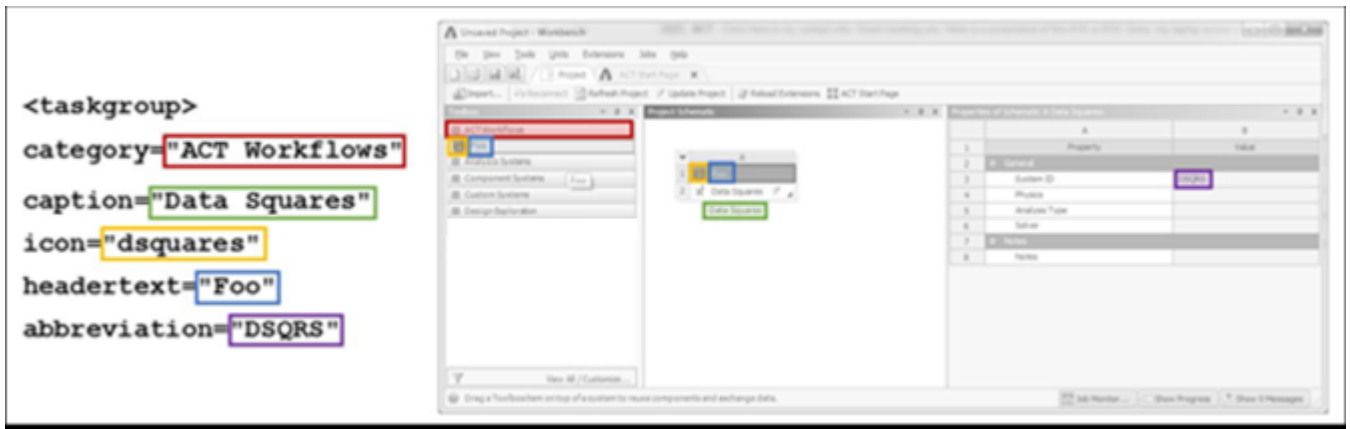


Defining a Task Group

A *task group* is a collection of tasks defined in the element `<taskgroups>`. An individual task group is defined in the child element `<taskgroup>`. At minimum, each task group definition must include the attributes **name** and **version**.

```
<extension>
  <workflow name="MyWorkflow" context="Project" version="1">
    <taskgroups>
      <taskgroup name="DataSquares" ... headertext="Foo" />
    </taskgroups>
  </workflow>
</extension>
```

The following figure maps the preceding example to the Workbench interface.



Task group entries support standard ACT attributes. When creating custom task groups with ANSYS-installed Mechanical tasks, you can use attributes to specify additional properties.

```
<extension>
  <workflow name="MyWorkflow" context="Project" version="1">
    <taskgroups>
      <taskgroup name="CustomStructural" caption="Custom Structural Group" icon="custom_structural" category="AC
      ...
      <attributes SolverType="ANSYS"
                  SystemName="Static Structural (ANSYS)"/>
    </taskgroup>
    </taskgroups>
  </workflow>
</extension>
```

For the following tasks, you find the solver type by referring to `$(AttributeCombination)`, where the second part of the Physics Type and Solver Name combination is the Solver Type.

- `SimulationModelCellTemplate`
- `SimulationSetupCellTemplate_$(AttributeCombination)`
- `SimulationSolutionCellTemplate_$(AttributeCombination)`
- `SimulationResultsCellTemplate_$(AttributeCombination)`

For a static structural ANSYS-based task, the `StructuralStaticANSYS` combination shows `ANSYS` as the Solver Type. You can obtain more combinations from `%AWP_ROOT202%\Addins\Simulation\AnalysisTemplates.xml`.

You use the child elements `<includetask>` and `<includeGroup>` to specify the one or more tasks or nested task groups to include. In the element `<includetask>`, you can reference both the custom tasks defined in your extension and ANSYS-installed tasks. Setting the optional attribute `external` to `true` specifies that the referenced task is defined outside of the extension.

For an ANSYS-installed task, the attribute `name` must be set to the template name for the given ANSYS component. Optionally, you can set the attribute `caption` to use your own display text. For a list of ANSYS-installed system with their components and component display names, see [Appendix B: ANSYS-Installed System Component Template and Display Names](#) (p. 187). To review examples of task groups referencing custom tasks, see [Generic Material Transfer](#) (p. 103) or [Generic Mesh Transfer](#) (p. 99).

If you define a caption, it overrides the task-level caption. Otherwise, the task group-level caption is used.

The basic structure of a task group definition follows.

```
...
<taskgroups>
  <taskgroup name="" caption="" icon="" category="" abbreviation="" version="1" isparametricgroup="False">
    <includetask name="" external="" />
    <includeGroup name="" />
  </taskgroup>
</taskgroups>
...
```

Note:

At present, ACT-defined tasks cannot be added to ANSYS-installed task groups. Also, nested task groups within a task group are not recognized by Workbench.

IronPython Script for a Custom Workflow

IronPython scripts define the actions to be executed during a callback invocation. These scripts execute within the Workbench Python interpreter. As a result, scripts have full access to the scriptable Workbench API. For more information, see [Using Journals and Scripts](#) in the *Discovery AIM and Workbench Scripting Guide*.

For example, the script is used to create update instructions for producing and consuming data. If any task produces or consumes data, you must supply an update routine that processes input and output types as declared by the workflow definition. For a table of supported task inputs and outputs, see [Appendix A: Component Input and Output Tables](#) (p. 111).

The following topics describe inputs, outputs, and convenience APIs:

[Upstream Data Consumption \(Input\)](#)

[Data Generation \(Output\)](#)

[Convenience APIs](#)

Upstream Data Consumption (Input)

Typically, tasks need to implement complex source handling logic, connection tracking routines, and a refresh procedure to consume data. ACT abstracts these complexities by automatically performing these actions during refresh. It obtains upstream data and stores it in a dictionary accessible during the update of the task. The task can obtain this data by calling [Convenience APIs](#) (p. 84).

Data Generation (Output)

Tasks that produce output data (for example, declare output types in the task group definition) must ensure that their custom update routine assigns output data.

The XML file prepares empty data objects representing task outputs. You must only set the correct transfer properties that downstream consumers interrogate. To determine which properties you must set, see [Appendix C: Data Transfer Types \(p. 209\)](#).

For example, a material transfer to a downstream task **Engineering Data** must set the property **DataReference TransferFile** on a MatML31 data object to the file reference of a registered matml-formatted XML file, all completed during the update routine.

Convenience APIs

Convenience APIs are IronPython queries that provide simple access to task-stored input and output data. The available convenience APIs are:

GetInputDataByType

Returns a **List<object>** containing upstream data for a given type. For example:

```
upstreamData =  
    container.GetInputDataByType( InputType="MeshingMesh" )  
meshFileRef = None  
upstreamDataCount = upstreamData.Count  
if upstreamDataCount > 0:  
    meshFileRef = upstreamData[0]
```

GetOutputData

Returns a **Dictionary<string, List<DataReference>>** holding the task's output types. For example:

```
outputRefs = container.GetOutputData()  
meshOutputSet = outputRefs["SimulationGeneratedMesh"]  
meshOutput = meshOutputSet[0]  
meshOutput.TransferFile = meshFileRef
```

GetCustomEntity

Obtains the custom data entity from a container based on the entity's name and type. If the name and type are not specified, the default ACT object is returned. This object contains the properties defined by the workflow task from the elements **<propertygroup>** and **<property>**.

```
entity = ACT.GetCustomEntity(container)
```

GetCustomEntityPropertyValue

This query returns the value of a property defined on a custom data entity.

```
inputValue = ACT.GetCustomEntityPropertyValue(entity, "Input")
```

Users can also directly access properties off of a task object.

SetCustomEntityPropertyValue

This command handles the setting of a custom entity's property value.

```
ACT.SetCustomEntityPropertyValue(entity, "Output", outputValue)
```

Users can also directly set properties off of a task object.

GetTaskByName

This command accesses the data model to facilitate task searches. A usage example appears in the figure at the end of this section.

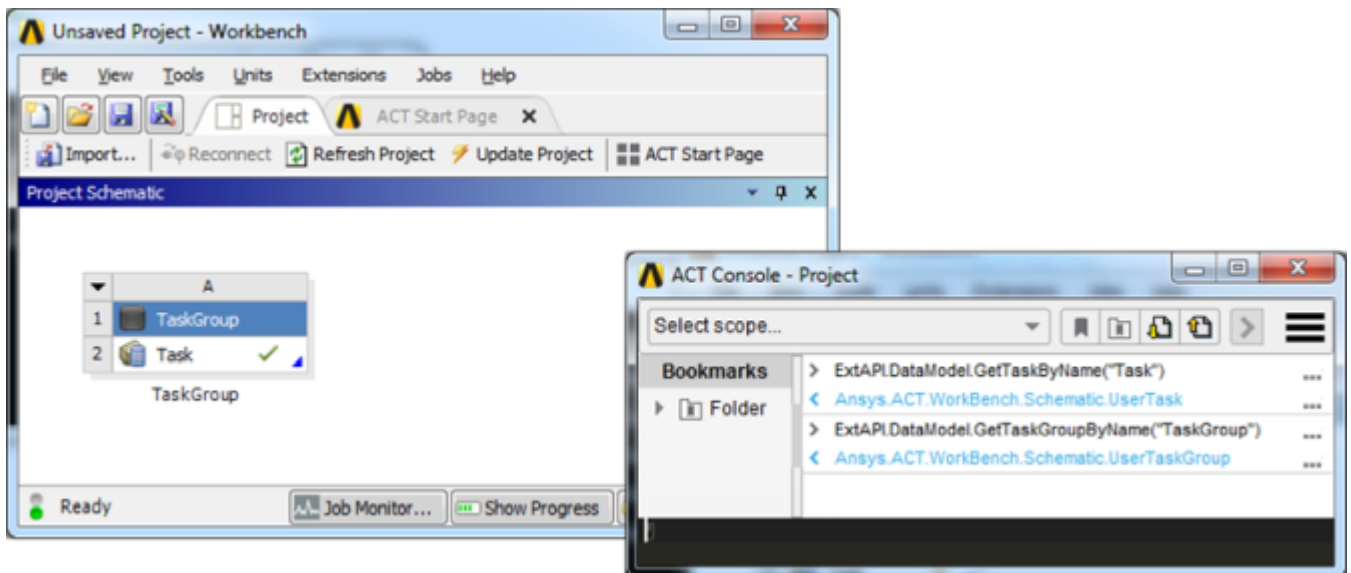
```
ExtAPI.DataModel.GetTaskByName(taskName)
```

GetTaskGroupName

This command accesses the data model to facilitate task group searches. A usage example appears in the figure at the end of this section.

```
ExtAPI.DataModel.GetTaskGroupName(taskGroupName)
```

The following figure shows convenience queries for tasks and task groups.



Simulation Workflow Integration Examples

This section describes [supplied extensions](#) that provide examples for custom workflows in the Workbench **Project Schematic**:

[Parametric Task Group](#)

[External Application Integration with Parameter Definition](#)

[External Application Integration with Custom Data and Remote Job Execution](#)

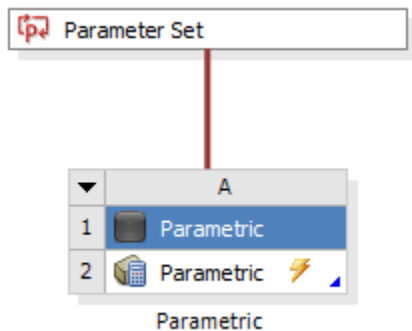
[Generic Mesh Transfer](#)

[Custom Transfer](#)

[Generic Material Transfer](#)

Parametric Task Group

The supplied extension **Parametric** demonstrates the creation of a parametric task group using the attribute **isparametricgroup**. When set to **true**, this attribute indicates that the task group operates only on design points. As such, the task group is added below the **Parameter Set** bar. The focus on parameters in this extension enables you to incorporate DesignXplorer-like functionality.



Creating the Extension for Creating a Parametric Task Group

The file `Parametric.xml` follows.

```
<extension version="1" name="Parametric">
  <guid shortid="Parametric">69d0095b-e138-4841-a13a-de12238c83f5</guid>
  <script src="parametric.py" />
  <interface context="Project">
    <images>images</images>
  </interface>
  <workflow name="wf2" context="Project" version="1">
    <tasks>
      <task name="Parametric" caption="Parametric" icon="parametric_component" version="1">
        <callbacks>
          <onupdate>update</onupdate>
        </callbacks>
      </task>
    </tasks>
  </workflow>
</extension>
```

```

        <inputs>
            <input/>
        </inputs>
        <outputs/>
    </task>
</tasks>
<taskgroups>
    <taskgroup name="Parametric" caption="Parametric" icon="parametric" category="ACT Custom Workflows">
        <includeTask name="Parametric" caption="Parametric"/>
    </taskgroup>
</taskgroups>
</workflow>
</extension>

```

This XML file performs the following actions:

- References the script `parametric.py`.
- Defines a single task in the element `<tasks>`.
- Defines the inputs and outputs in the elements `<inputs>` and `<outputs>`. Note that an empty input is defined.
- Defines a single task group with a single task in the element `<taskgroups>`. Note that the attribute `isparametricgroup` is set to `true`.

Defining Functions for Creating a Parametric Task Group

The IronPython script `parametric.py` performs an update on the parameters. Because the XML file has the callback `<onupdate>`, the method `update` is defined in the script.

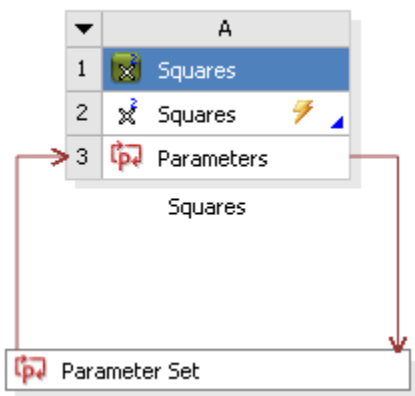
```

def update(task):
    ExtAPI.Log.WriteMessage("test")

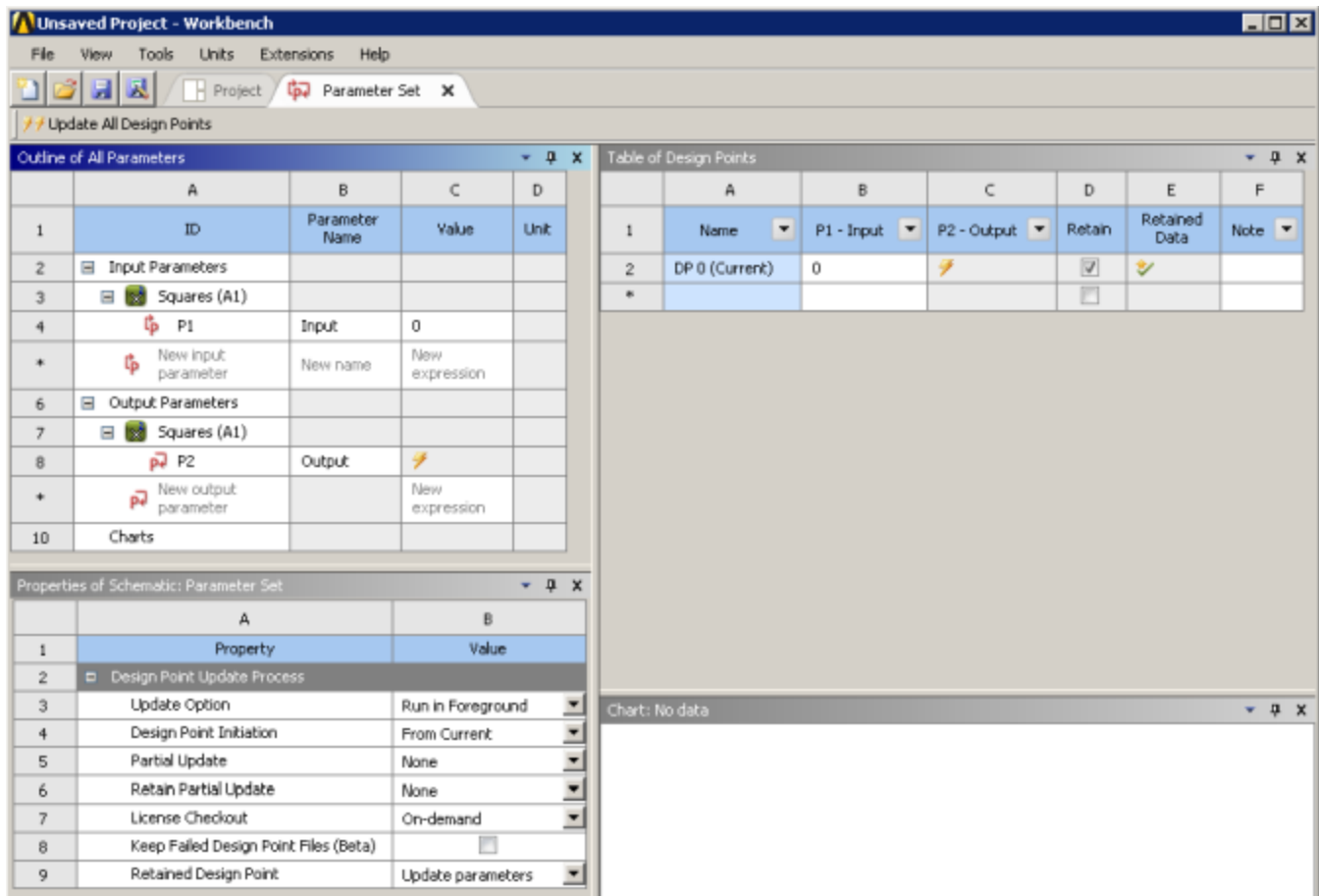
```

External Application Integration with Parameter Definition

The supplied extension **Squares** demonstrates the use of parameter definitions to integrate a lightweight external application. Driven by parameters defined in the XML file, the external application squares the value of an input number, which is displayed in the **Parameter Set** bar.



The external application updates the output parameter to the computed square value. The use of parameters in this example enables you to leverage DesignXplorer functionality.



Creating the Extension for Integrating an External Application with Parameter Definition

The file `Squares.xml` follows.

```
<extension version="1" name="Squares">
  <guid shortid="Squares">69d0095b-e138-4841-a13a-de12238c83f4</guid>
  <script src="Squares.py" />
  <interface context="Project">
    <images>images</images>
  </interface>
  <workflow name="wf1" context="Project" version="1">
    <tasks>
      <task name="Squares" caption="Squares" icon="squares_component" version="1">
        <callbacks>
          <onupdate>update</onupdate>
        </callbacks>
        <inputs>
          <input/>
        </inputs>
        <outputs/>
        <parameters>
          <parameter name="Input" caption="Input" usage="Input" control="Double" version="1"/>
          <parameter name="Output" caption="Output" usage="Output" control="Double" version="1"/>
        </parameters>
      </task>
    </tasks>
  </workflow>
</extension>
```

```
<taskgroups>
  <taskgroup name="Squares" caption="Squares" icon="squares" category="ACT Custom Workflows" abbreviat
    <includeTask name="Squares" caption="Squares"/>
  </taskgroup>
</taskgroups>
</workflow>
</extension>
```

This XML file performs the following actions:

- References the IronPython script `Squares.py`.
- Defines a single task in the element **<tasks>**.
- Defines the inputs and outputs in the elements **<inputs>** and **<outputs>**. Also defines an empty input.
- Defines two parameters in the element **<parameters>**.
- Defines a single task group with a single task in the element **<taskgroups>**.

Defining Functions for Integrating an External Application with Parameter Definition

The IronPython script `Squares.py` follows.

```
import System

import clr
clr.AddReference("Ans.Utilities")
import Ansys.Utilities

#convenience method to look up parameters based on a predetermined ID
def GetParameterByName(parameters, id):
    match = None
    for param in parameters:
        if param.ParameterName == id:
            match = param
            break
    return match

def update(task):
    container = task.InternalObject
    context = ExtAPI.DataModel.Context
    activeDir = container.GetActiveDirectory()
    extensionDir = ExtAPI.ExtensionManager.CurrentExtension.InstallDir
    exeName = "ExampleAddinExternalSolver.exe"
    solverPath = System.IO.Path.Combine(extensionDir, exeName)

    #get parameters owned by container

    params = None
    lock = context.ContainerReadLock(container)
    params = context.Project.GetDataReferencesByType(container, "ParameterAdapter")
    lock.Dispose()

    #isolate specific parameters

    inputParam = GetParameterByName(params, "Input")
    outputParam = GetParameterByName(params, "Output")

    #prep i/o file paths
```

```

inputFileName = "input.txt"
outputFileName = "output.txt"
dpInputFile = System.IO.Path.Combine(activeDir, inputFileName)
dpOutputFile = System.IO.Path.Combine(activeDir, outputFileName)

#write input file

if inputParam != None and outputParam != None:
    val = inputParam.Value
    #write input file
    f = open(dpInputFile, "w")
    f.write('input='+val.ToString(System.Globalization.NumberFormatInfo.InvariantInfo))
    f.close()

#run exe

runInMono = Ansys.Utilities.ApplicationConfiguration.DefaultConfiguration.IsRuntimeMono
monoPath = "mono"
monoArgs = System.String.Format("{0} \"{1}\" \"{2}\"", solverPath, dpInputFile, dpOutputFile)
info = None
if runInMono:
    info = System.Diagnostics.ProcessStartInfo(monoPath, monoArgs)
else:
    info = System.Diagnostics.ProcessStartInfo(solverPath, System.String.Format("\"{0}\" \"{1}\"",
dpInputFile, dpOutputFile))
    info.CreateNoWindow = True
    info.WindowStyle = System.Diagnostics.ProcessWindowStyle.Minimized
p = System.Diagnostics.Process.Start(info)
p.WaitForExit()

#read output file

outValue = None
f = open(dpOutputFile, "r")
currLine = f.readline()
while currLine != "":
    valuePair = currLine.split('=')
    outValue = System.Int32.Parse(valuePair[1], System.Globalization.NumberFormatInfo.InvariantInfo)
    currLine = f.readline()
f.close()

#set output value

if outValue == None:
    raise Exception("Error in update - no output value detected!")
else:
    outputParam.Value = outValue

```

This script performs the following actions:

- Obtains the parameters.
- Prepares the inputs.
- Writes the input file.
- Runs the external solver.
- Reads the output file.
- Sets the parameters to the calculated solver values.

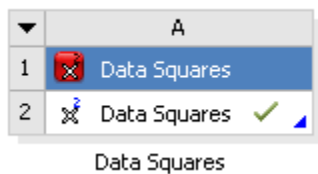
Because the XML file has the callback **<onupdate>**, the method **update** is defined in the script. All ACT workflow callbacks get a **container** for the task and a **context**.

External Application Integration with Custom Data and Remote Job Execution

The supplied extension **DataSquares** demonstrates the use of custom task properties to integrate a lightweight external application. This extension specifies that the defined task should be submitted via ANSYS Remote Solve Manager (RSM) for remote job execution.

Driven by the custom task properties defined in the XML file, this external application squares the value of an input number, which it displays in the **Parameter Set** bar. Because the task defines a RSM job, the task update can either be run locally or the calculations can be submitted via RSM for remote job execution. Once the job has completed, Workbench merges the job back into the active session and retrieves results. The custom task defined in the extension then updates the custom property values to the externally computed square value.

This extension also demonstrates progress and project reporting functionality. The following figure shows the **Data Squares** task group in the **Project Schematic**.



Creating the Extension for Integrating an External Application with Custom Data

The file `DataSquares.xml` follows.

```
<extension version="1" name="DataSquares">
  <guid shortid="DataSquares">69d0095b-e138-4841-a13a-de12238c83f2</guid>
  <script src="datasquares_complete.py" />
  <interface context="Project">
    <images>images</images>
  </interface>
  <workflow name="MyWorkflow" context="Project" version="1">
    <tasks>
      <task name="DataSquares" caption="Data Squares" icon="dsquares_component" version="1">
        <callbacks>
          <onupdate>update</onupdate>
          <onstatus>status</onstatus>
          <onreport>report</onreport>
          <onreset>reset</onreset>
        </callbacks>
        <propertygroup name="Inputs">
          <property name="Input" caption="Input" control="double" default="0.0" readonly="False" needu
        </propertygroup>
        <propertygroup name="Outputs">
          <property name="Output" caption="Output" control="double" default="0.0" readonly="True" visi
        </propertygroup>
        <inputs>
          <input/>
        </inputs>
        <outputs/>
        <rsmjob name="squares" deletefiles="True" version="1">
          <inputfile id="1" name="input1.txt"/>
          <outputfile id="1" name="output1.txt"/>
          <program>
            <platform name="Win64" path="%AWP_ROOT171%\Addins\ACT\extensions\DataSquares\ExampleAddi
```

```

        <platform name="Linux64" path="%AWP_ROOT171%/Addins/ACT/extensions/DataSquares/ExampleAd
        <argument name=" " value="inputFile:1" separator=" " />
        <argument name=" " value="outputFile:1" separator=" " />
    </program>
    <callbacks>
        <oncreatejobinput>createJobInput</oncreatejobinput>
        <onjobstatus>getJobStatus</onjobstatus>
        <onjobcancellation>cancelJob</onjobcancellation>
        <onjobreconnect>reconnectJob</onjobreconnect>
    </callbacks>
    </rsmjob>
</task>
</tasks>
<taskgroups>
    <taskgroup name="DataSquares" caption="Data Squares" icon="dsquares" category="ACT Custom Workflows">
        <includeTask name="DataSquares" caption="Data Squares" />
    </taskgroup>
</taskgroups>
</workflow>
</extension>

```

This XML file performs the following actions:

- References the script `datasquares_complete.py`.
 - Defines a single task in the element **<tasks>**. Within this element:
 - Defines the callback **<onreport>**, which is called when a project report is generated. This allows the task to access the report object and add its own task-specific reporting content.
 - Defines two property groups with the element **<propertygroup>**. Within the two property groups, **<property>** tags define the properties **Input** and **Output**.
 - Defines the inputs and outputs in the elements **<inputs>** and **<outputs>**. Note that an empty input is defined here, although this is not required.
 - Specifies via the element **<rsmJob>** that the task can be sent via RSM as a job for remote execution. To specify RSM support, you add the element **<rsmJob>** to the task in the existing XML extension definition file. In the RSM job specification:
 - Attributes specify the name and indicate the files to manage upon successful completion or cancellation of the job.
 - The tag **<platform>** specifies a supported platform.
 - The tag **<argument>** specifies the external application's command line parameters, including input and output file paths.
- When using referenced file paths, the same **ID** can be used across categories but not within categories. An input and an output can both have an **ID** of 1, but two inputs cannot have an **ID** of 1. During the execution, all input and output file names are transformed into fully qualified file paths.
- The element **<program>** informs the extension about the task's executable, supported platforms, and corresponding command line arguments. Environment variables can be used for the executable path and previously defined arguments can be referenced by **ID**.

→ The callbacks invoke methods defined in the script to create the job, check job status, provide the capability to cancel the job, and access output files and results.

- Defines a single task group in the element **<taskgroups>** that includes the task **DataSquares**

The following callbacks reference methods in the IronPython script. The first three callbacks invoke standard task actions. The subsequent callbacks invoke actions related to the RSM job specification.

<onupdate>

Invokes the method **update**. Called when the user updates the task.

<onstatus>

Invokes the method **status**. Called when the product (such as Workbench) asks the task for its current state.

<onreport>

Invokes the method **report**. Called when the user generates a project report. This allows the extension to access the report object and add their own task-specific reporting content.

<oncreatejobinput>

Invokes the method **createJobInput**. Called when the RSM job requires the generation of all specified input files. Must prepare and generate the input files specified for the RSM job.

As a method argument, ACT passes in the fully qualified input file paths. For this example, only one input file is generated. It contains the input value for the squares solver. ACT retrieves the first entry from the input file list, opens the file, writes the input value line, and closes the file. ACT requires no further action.

<onjobstatus>

Invokes the method **getJobStatus**. Must tell Workbench if the job has finished. Polled by Workbench several times during the RSM job execution to determine if the update execution has completed. ACT supplies the fully qualified output file list as a method argument.

For the example, **True** is returned when the presence of the solver's sole output file is detected. Otherwise, **False** is returned.

<onjobcancellation>

Invokes the method **cancelJob**. Called when the remote update is stopped before the job completes. Must conduct any clean-up actions required due to a user-initiated stoppage. ACT supplies the fully qualified lists of both input and output files as method arguments.

For this example, no other action is performed. Only file cleanup is required. Setting the method **DeleteFiles** to **True** automatically handles the file cleanup.

<onjobreconnect>

Invokes the method **reconnectJob**. Called when the job status reports a completed job. Must retrieve remotely solved information from output files and update the task's data, rendering it up-to-date.

For this example, the output file generated by the solver is read and the output property that is defined on the task is updated.

The next topic provides additional information about these methods.

Defining Functions for Integrating an External Application with Custom Data

The IronPython script `dataqaures_complete.py` defines the methods that are invoked by callbacks in the XML file. The methods `update`, `status`, and `report` are standard methods defined for the task. The methods `createJobInput`, `getJobStatus`, `cancelJob`, and `reconnectJob` are related to the RSM job specification.

```
import clr
clr.AddReference("Ans.Core")

def update(task):
    container = task.InternalObject
    context = ExtAPI.DataModel.Context
    activeDir = container.GetActiveDirectory()
    extensionDir = ExtAPI.ExtensionManager.CurrentExtension.InstallDir
    exeName = "ExampleAddinExternalSolver.exe"
    solverPath = System.IO.Path.Combine(extensionDir, exeName)

    monitor = context.ProgressMonitor
    monitor.BeginTask("Data Sqaures Solver", 3, None)
    monitor.TaskDetails = "Preparing solver input..."
    System.Threading.Thread.Sleep(2000)
    monitor.UpdateTask(1, None)

    #get param values
    inputValue = task.Properties["Inputs"].Properties["Input"].Value

    #prep i/o file paths

    inputFileName = "input.txt"
    outputFileName = "output.txt"
    dpInputFile = System.IO.Path.Combine(activeDir, inputFileName)
    dpOutputFile = System.IO.Path.Combine(activeDir, outputFileName)

    #write input file
    f = open(dpInputFile, "w")
    f.write('input='+inputValue.ToString(System.Globalization.NumberFormatInfo.InvariantInfo))
    f.close()

    monitor.UpdateTask(1, None)

    monitor.TaskDetails = "Executing Solver..."
    System.Threading.Thread.Sleep(2000)

    #run exe

    runInMono = Ansys.Utilities.ApplicationConfiguration.DefaultConfiguration.IsRuntimeMono
    monoPath = "mono"
    monoArgs = System.String.Format("{0} \"{1}\" \"{2}\"", solverPath, dpInputFile, dpOutputFile)
    info = None
    if runInMono:
        info = System.Diagnostics.ProcessStartInfo(monoPath, monoArgs)
    else:
        info = System.Diagnostics.ProcessStartInfo(solverPath, System.String.Format("\"{0}\" \"{1}\"",
dpInputFile, dpOutputFile))
        info.CreateNoWindow = True
        info.WindowStyle = System.Diagnostics.ProcessWindowStyle.Minimized
        p = System.Diagnostics.Process.Start(info)
        p.WaitForExit()

    monitor.UpdateTask(1, None)

    monitor.TaskDetails = "Retrieving results from solver..."
    System.Threading.Thread.Sleep(2000)

    #read output file

    outputValue = None
```

```

f = open(dpOutputFile, "r")
currLine = f.readline()
while currLine != "":
    valuePair = currLine.split('=')
    outputValue = System.Double.Parse(valuePair[1], System.Globalization.NumberFormatInfo.InvariantInfo)
    currLine = f.readline()
f.close()

monitor.UpdateTask(1, None)

#set output value

if outputValue == None:
    raise Exception("Error in update - no output value detected!")
else:
    task.Properties["Outputs"].Properties["Output"].Value = outputValue
monitor.TaskDetails = "Solve completed..."
System.Threading.Thread.Sleep(2000)
monitor.EndTask(None)

def createJobInput(task, inputFilePaths):
    ExtAPI.Log.WriteMessage('creating job input')
    inputFilePath = inputFilePaths[0]
    #get param values
    inputValue = task.Properties["Inputs"].Properties["Input"].Value

    #write input file
    ExtAPI.Log.WriteMessage("Writing input value (" + str(inputValue) + ") to file (" + inputFilePath + ")")
    f = open(inputFilePath, "w")
    f.write('input=' + inputValue.ToString(System.Globalization.NumberFormatInfo.InvariantInfo))
    f.close()
def reconnectJob(task, outputFilePaths):
    ExtAPI.Log.WriteMessage('reconnecting job')
    outputValue = None
    outputFilePath = outputFilePaths[0] #I know we only have one specified based on our definition...so work off
    f = open(outputFilePath, "r")
    currLine = f.readline()
    while currLine != "":
        valuePair = currLine.split('=')
        outputValue = System.Double.Parse(valuePair[1], System.Globalization.NumberFormatInfo.InvariantInfo)
        currLine = f.readline()
    f.close()
    #set output value
    ExtAPI.Log.WriteMessage("Retrieved value (" + str(outputValue) + ") from file (" + outputFilePath + ")")
    if outputValue == None:
        raise Exception("Error in update - no output value detected!")
    else:
        task.Properties["Outputs"].Properties["Output"].Value = outputValue
def getJobStatus(task, outputFiles):
    ExtAPI.Log.WriteMessage('checking job status')
    outputFilePath = outputFiles[0]
    finished = System.IO.File.Exists(outputFilePath)
    return finished
def cancelJob(task, inputFiles, outputFiles):
    #nothing to do...just print a message for now.
    ExtAPI.Log.WriteMessage('performing cancellation clean up')
import clr
clr.AddReference("Ans.ProjectSchematic")
clr.AddReference("ReportUtility.Interop")
import Ansys.ReportUtility.Interop
import Ansys.ProjectSchematic
def status(task):
    status = Ansys.ProjectSchematic.Queries.ComponentState(Ansys.ProjectSchematic.State.Unfulfilled, "This is un
    return None
def report(task, report):
    root = report.GetRootSection()
    section = Ansys.ReportUtility.Interop.ReportSection("My Custom ACT Task Report Content")
    text = Ansys.ReportUtility.Interop.ReportText("", "Sample text from the data squares component")
    section.AddChild(text)
    root.AddChild(section)
def reset(task):

```

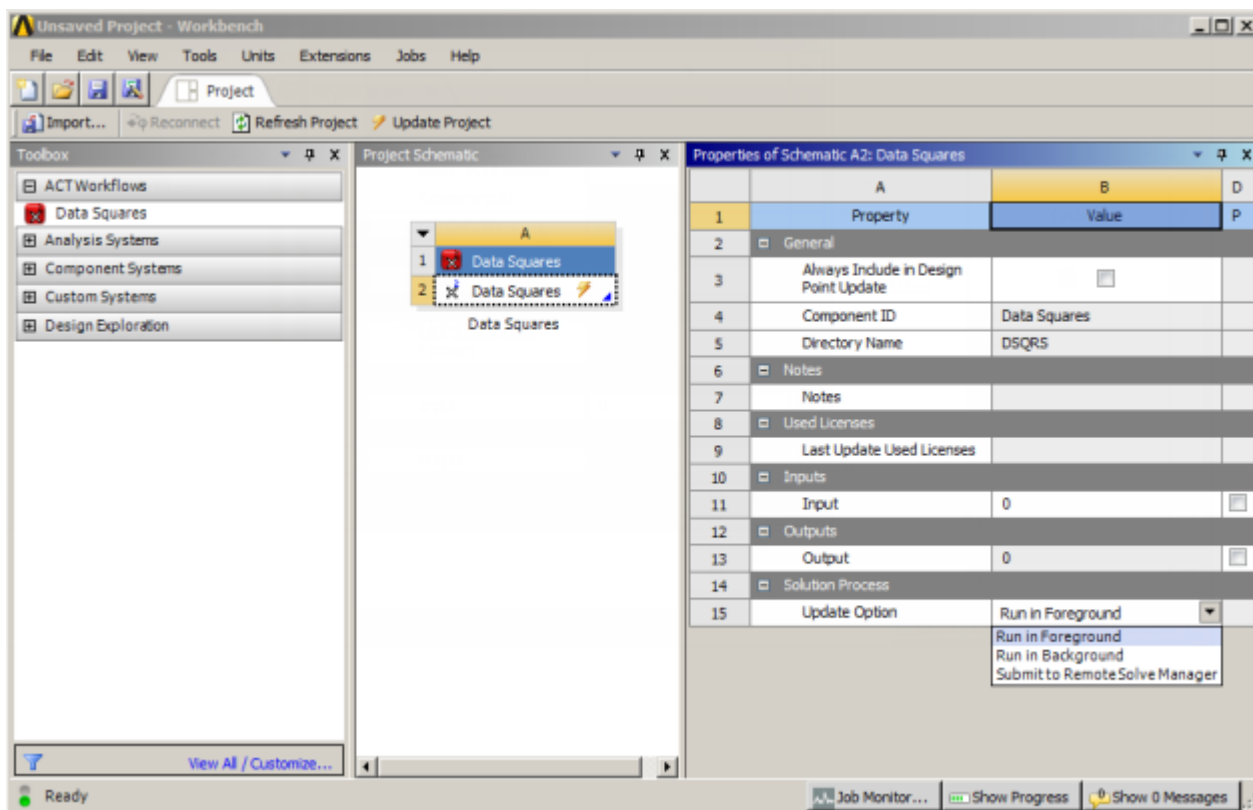
```
task.Properties["Inputs"].Properties["Input"].Value = 0
task.Properties["Outputs"].Properties["Output"].Value = 0
```

Using RSM Job Submission Capabilities

Once you've defined an extension with an RSM job specification, you can submit your task as an RSM job for remote execution:

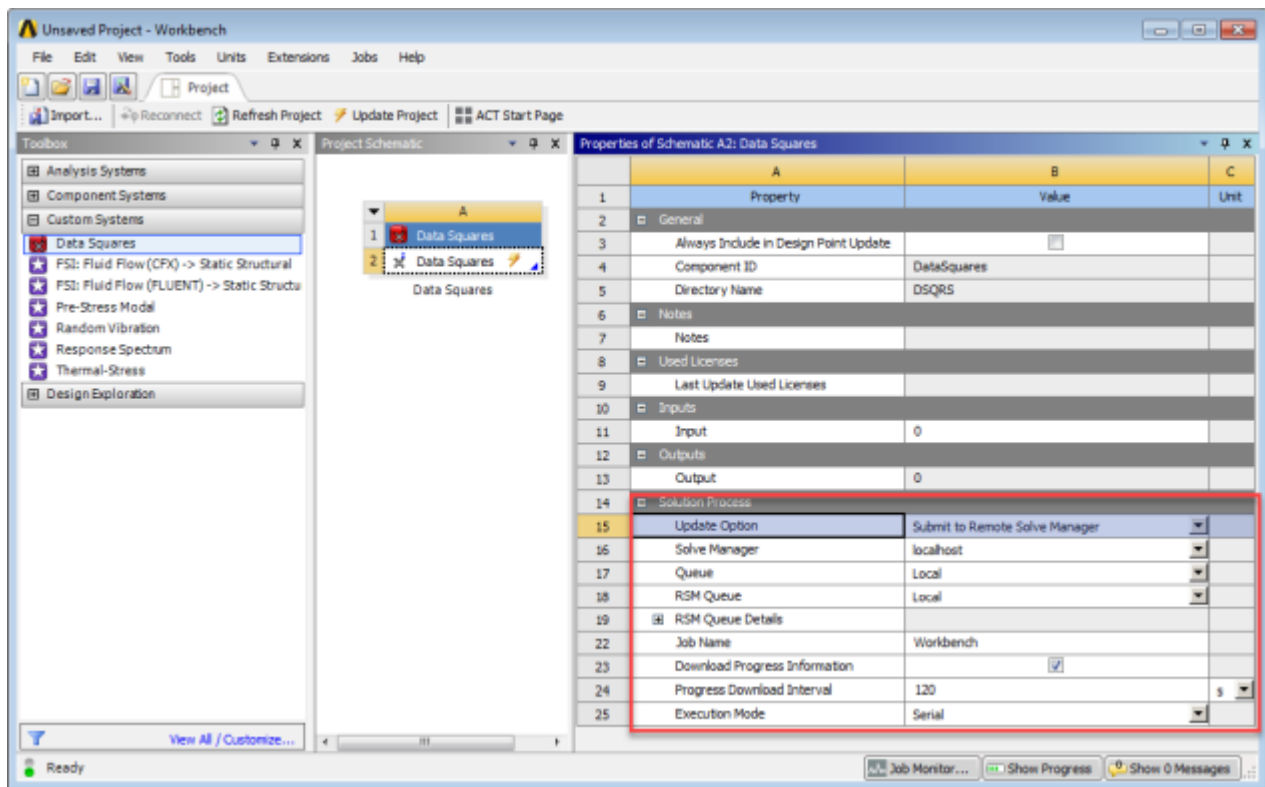
1. Install and load the extension to Workbench as usual.
2. Add a task group **Data Squares** in the **Project Schematic**.

Solution Process properties become available for the task group.



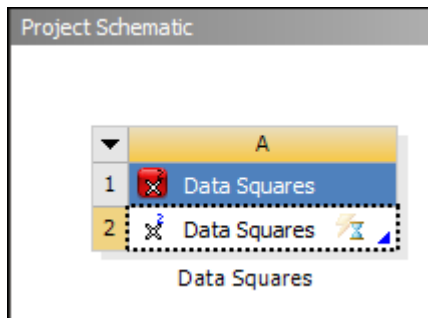
3. To specify that your task is to be executed remotely, set **Update Option** to **Submit to Remote Solve Manager**.

Additional **Solution Process** properties become available.



- Set **Solution Process** properties. Then, save and update your project.

The task submits its process to v as a new job. Once RSM accepts the job, the task transitions to the pending state.



- To view the status of your RSM job while it is being updated, open RSM.

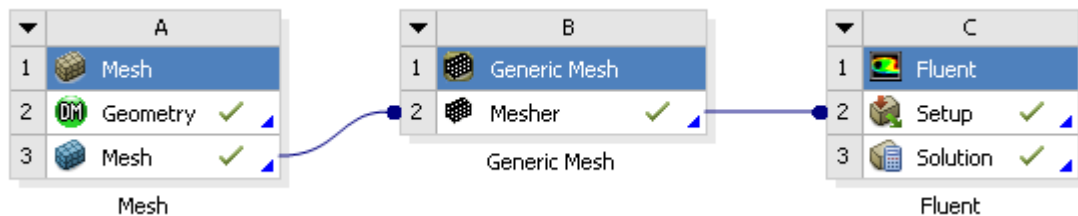
Note:

To run your jobs locally in the background, set **Update Option** to **Run in Background**. Updates are then treated in a similar manner to true remote updates. Workbench prepares the update and invokes the same callbacks as in the example, but the update proceeds as a separate local update instead of engaging RSM.

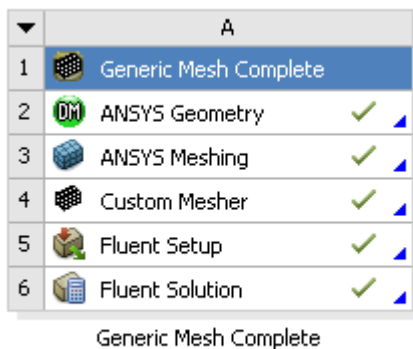
Generic Mesh Transfer

The supplied extension **GenericMeshTransfer** implements two custom mesh transfer task groups, each with consuming (downstream) and providing (upstream) connections.

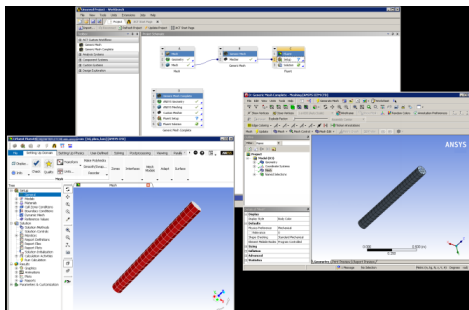
- The first custom task group references a single custom task. It consumes a mesh from an upstream ANSYS-installed task group and passes it to a downstream ANSYS-installed task group.



- The second custom task group references a custom task and multiple ANSYS-installed tasks. The custom task consumes a mesh from an upstream ANSYS-installed task and passes it to a downstream ANSYS-installed task.



This extension also demonstrates input and output specification, file management capabilities, and the default **Edit** context menu.



Creating the Extension for Generic Mesh Transfer

The file `GenericMeshTransfer.xml` follows.

```

<extension version="1" name="GenericMeshTransfer">
  <guid shortid="GenericMeshTransfer">69d0095b-e138-4841-a13a-de12238c83f7</guid>
  <script src="generic_mesh_transfer.py" />
  <interface context="Project">
    <images>images</images>
  </interface>
  <workflow name="wf6" context="Project" version="1">
    <tasks>
      <task name="Mesher" caption="Mesher" icon="GenericMesh_cell" version="1">
        <callbacks>
          <onupdate>update</onupdate>
          <onedit>edit</onedit>
        </callbacks>
        <inputs>
          <input format="FluentMesh" type="MeshingMesh" count="1"/>
        </inputs>
        <outputs>
          <output format="FluentMesh" type="SimulationGeneratedMesh"/>
        </outputs>
      </task>
    </tasks>
    <taskgroups>
      <taskgroup name="GenericMeshTransfer" caption="Generic Mesh" icon="GenericMesh" category="ACT Custom">
        <includeTask name="Mesher" caption="Mesher"/>
      </taskgroup>
      <taskgroup name="GenericMeshTransferComplete" caption="Generic Mesh Complete" icon="GenericMesh" category="ACT Custom">
        <includeTask external="True" name="GeometryCellTemplate" caption="ANSYS Geometry"/>
        <includeTask external="True" name="SimulationMeshingModelCellTemplate" caption="ANSYS Meshing"/>
        <includeTask name="Mesher" caption="Custom Mesher"/>
        <includeTask external="True" name="FluentSetupCellTemplate" caption="Fluent Setup"/>
        <includeTask external="True" name="FluentResultsCellTemplate" caption="Fluent Solution"/>
      </taskgroup>
    </taskgroups>
  </workflow>
</extension>

```

This file specifies the two custom task groups that are to appear in your custom **ACT Workflows** task group in the Workbench **Toolbox**. It performs the following actions:

- References the script `generic_mesh_transfer.py`.
- Defines a single task in the element **<tasks>**.
- Defines the callback **<onedit>** for the **Mesher** task, which automatically creates a default **Edit** context menu for the task.
- Defines an input and an output in the elements **<inputs>** and **<outputs>**.
 - The input has **type** set to **MesheringMesh**, indicating that the input data type is a mesh.
 - Both the input and output have **format** set to **FluentMesh**, specifying that the input and output files have the same **FluentMesh** format.
- Defines two custom task groups in the element **<taskgroups>**.
 - The first custom task group, **GenericMeshTransfer**, references the custom task **Mesher**. It consumes a mesh from an upstream **Mesher** task group and passes it to a downstream **Fluent** task group.
 - The second custom task group, **GenericMeshTransferComplete**, references the custom task **Material** and also, by using the attribute **external**, the ANSYS-installed tasks **ANSYS Geometry**, **ANSYS Meshing**, **Fluent Setup**, and **Fluent Solution**. The task **Material** consumes a

mesh from the upstream task **ANSYS Meshing** and passes it to the downstream task **Fluent Setup**.

Defining Functions for Generic Mesh Transfer

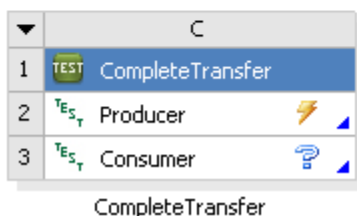
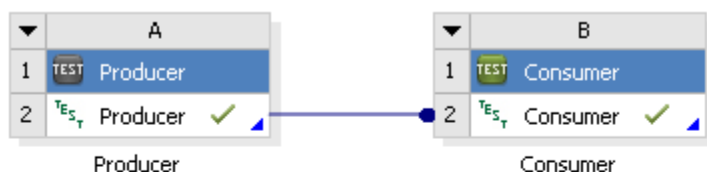
The IronPython script `generic_mesh_transfer.py` defines the functions for passing the mesh data to the downstream task group or task.

```
import clr
clr.AddReference("Ans.UI.Toolkit")
clr.AddReference("Ans.UI.Toolkit.Base")
import Ansys.UI.Toolkit

def update(task):
    container = task.InternalObject
    ExtAPI.Log.WriteMessage('in generic_mesh_transfer.py update method')
    #obtain input data
    upstreamData = container.GetInputDataByType(InputType="MeshingMesh")
    meshFileRef = None
    upstreamDataCount = upstreamData.Count
    if upstreamDataCount > 0:
        meshFileRef = upstreamData[0]
        #set our output so that we are just a pass through.
        outputRefs = container.GetOutputData()
        meshOutputSet = outputRefs["SimulationGeneratedMesh"]
        meshOutput = meshOutputSet[0]
        #meshOutput.MeshFile = meshFileRef
        meshOutput.TransferFile = meshFileRef
        ExtAPI.Log.WriteMessage(str(meshFileRef))
    #if no new data...nothing to process from upstream sources.
def edit(task):
    Ansys.UI.Toolkit.MessageBox.Show("Test!")
```

Custom Transfer

The supplied extension **CustomTransfer** implements a transfer from a "producing" custom task group to a consuming custom task group, creating connections between them (no ANSYS-installed products). It also demonstrates the creation of a custom task group with a single task rather than multiple tasks.

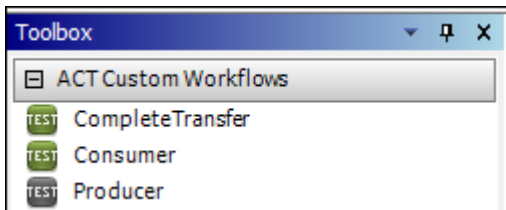


Creating the Extension for Custom Transfer

The file `CustomTransfer.xml` follows.

```
<extension version="1" name="CustomTransfer">
  <guid shortid="CustomTransfer">69d0095b-e138-4841-a13a-de12238c83f3</guid>
  <script src="customtransfer.py" />
  <interface context="Project">
    <images>images</images>
  </interface>
  <workflow name="wf4" context="Project" version="1">
    <tasks>
      <task name="Producer" caption="Producer" icon="test_component" version="1">
        <callbacks>
          <onupdate>producer_update</onupdate>
        </callbacks>
        <inputs>
          <input/>
        </inputs>
        <outputs>
          <output format="" type="MyData"/>
        </outputs>
      </task>
      <task name="Consumer" caption="consumer" icon="test_component" version="1">
        <callbacks>
          <onupdate>consumer_update</onupdate>
        </callbacks>
        <inputs>
          <input/>
          <input format="" type="MyData"/>
        </inputs>
        <outputs/>
      </task>
    </tasks>
    <taskgroups>
      <taskgroup name="Producer" caption="Producer" icon="producer_system" category="ACT Custom Workflows">
        <includeTask name="Producer" caption="Producer"/>
      </taskgroup>
      <taskgroup name="Consumer" caption="Consumer" icon="consumer_system" category="ACT Custom Workflows">
        <includeTask name="Consumer" caption="Consumer"/>
      </taskgroup>
      <taskgroup name="CompleteTransfer" caption="CompleteTransfer" icon="consumer_system" category="ACT C">
        <includeTask name="Producer" caption="Producer"/>
        <includeTask name="Consumer" caption="Consumer"/>
      </taskgroup>
    </taskgroups>
  </workflow>
</extension>
```

This XML file defines custom task groups named **Producer**, **Consumer**, and **CompleteTransfer**, all of which appear in the Workbench **Toolbox**.



It performs the following actions:

- References the IronPython script `customtransfer.py`.

- Defines two tasks in the element **<tasks>**: **Producer** and **Consumer**.
- Defines three task groups in the element **<taskgroups>**: **Producer**, **Consumer**, and **CompleteTransfer**. The task groups **Producer** and **Consumer** each contain a single task. The task group **CompleteTransfer** contains two tasks.

Defining Functions for Custom Transfer

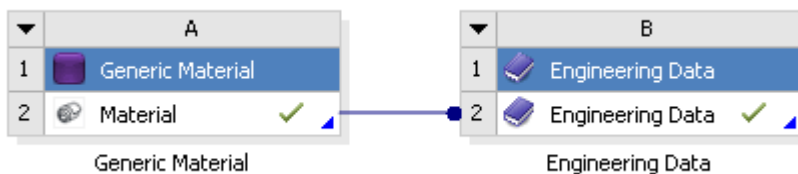
The IronPython script `customtransfer.py` defines the functions that provide update instructions for the producing task group and for the consuming task to obtain the output data from the upstream producer.

```
def consumer_update(task):
    container = task.InternalObject
    #obtain input data
    upstreamData = container.GetInputDataByType( InputType="MyData" )
    fileRef = None
    upstreamDataCount = upstreamData.Count
    if upstreamDataCount > 0:
        fileRef = upstreamData[0]
        AssociateFileWithContainer(fileRef, container)
        ExtAPI.Log.WriteMessage("Recieved file "+fileRef.Location+" from producer.")
    #if no new data...nothing to process from upstream sources.
def producer_update(task):
    container = task.InternalObject
    extensionDir = ExtAPI.ExtensionManager.CurrentExtension.InstallDir
    filePath = System.IO.Path.Combine(extensionDir, "Sample_Materials.xml")
    fileRef = None
    isRegistered = IsFileRegistered(FilePath=filePath)
    if isRegistered == True:
        fileRef = GetRegisteredFile(filePath)
    else:
        fileRef = RegisterFile(FilePath=filePath)
        AssociateFileWithContainer(fileRef, container)
    outputRefs = container.GetOutputData()
    outputSet = outputRefs["MyData"]
    myData = outputSet[0]
    myData.TransferFile = fileRef
```

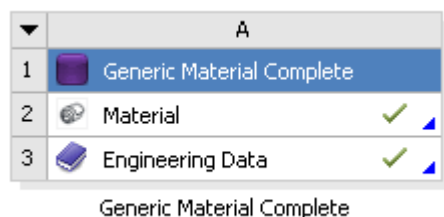
Generic Material Transfer

The supplied extension **GenericMaterialTransfer** implements two custom task groups, each passing material data to a downstream task group or task.

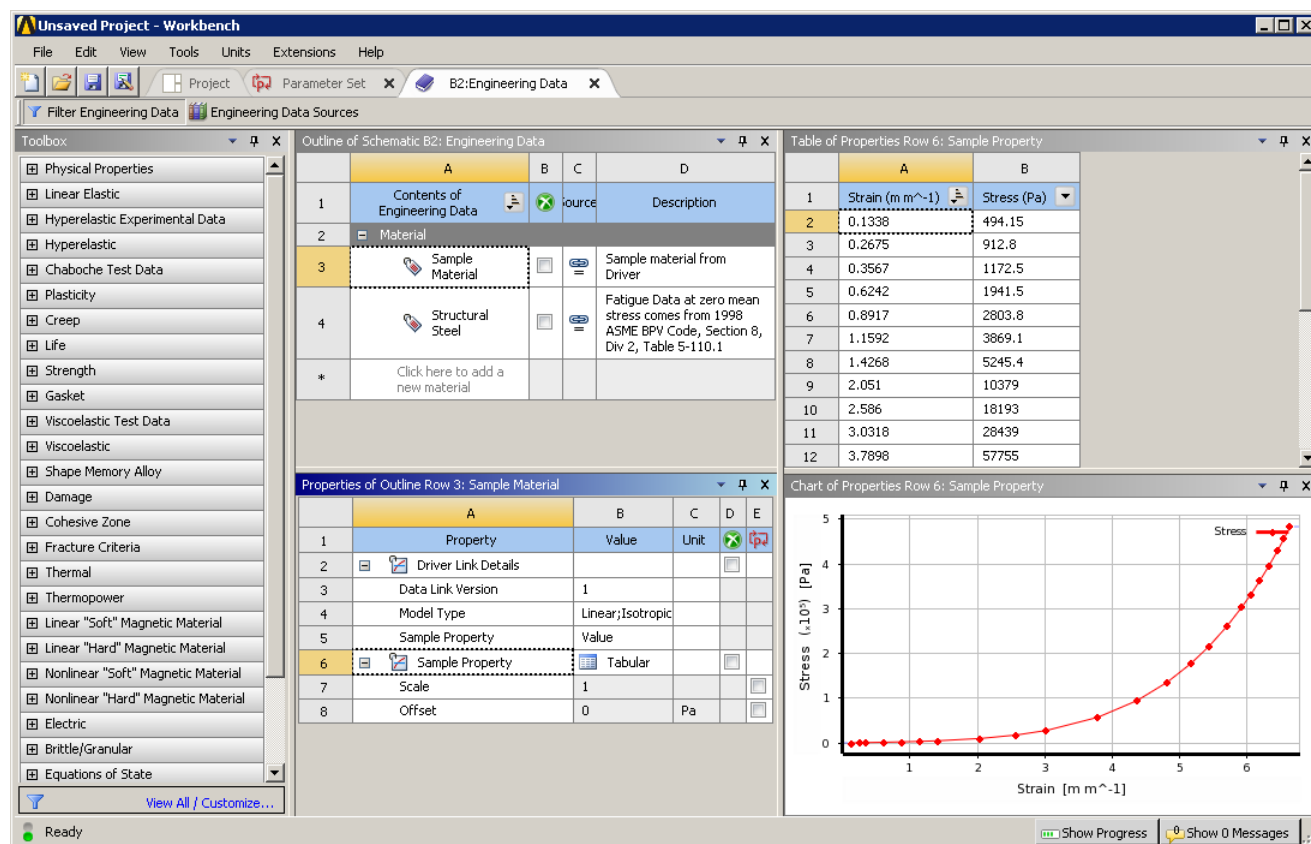
- The first custom task group references a single custom task, **Material**, that passes MatML-formatted material data to a downstream ANSYS-installed task group, **Engineering Data**.



- The second custom task group references both a custom task and an ANSYS-installed task. The custom task, **Material**, passes MatML-formatted material data to the downstream ANSYS-installed task, **Engineering Data**.



This extension also demonstrates input and output specification and file management capabilities.



Creating the Extension for Generic Material Transfer

The file `GenericMaterialTransfer.xml` follows.

```
<extension version="1" name="GenericMaterialTransfer">
  <guid shortid="GenericMaterialTransfer">69d0095b-e138-4841-a13a-de12238c83f8</guid>
  <script src="generic_material_transfer.py" />
  <interface context="Project">
    <images>images</images>
  </interface>
  <workflow name="wf5" context="Project" version="1">
    <tasks>
      <task name="Material" caption="Material" icon="material_cell" version="1">
        <callbacks>
          <onupdate>update</onupdate>
        </callbacks>
        <inputs>
          <input />
        </inputs>
        <outputs>
```

```

        <output format="" type="MatML31"/>
    </outputs>
</task>
</tasks>
<taskgroups>
    <taskgroup name="GenericMaterialTransfer" caption="Generic Material" icon="material_system" category="Generic Material">
        <includeTask name="Material" caption="Material"/>
    </taskgroup>
    <taskgroup name="GenericMaterialTransferComplete" caption="Generic Material Complete" icon="material_system">
        <includeTask name="Material" caption="Material"/>
        <includeTask external="True" name="EngDataCellTemplate" caption="Engineering Data"/>
    </taskgroup>
</taskgroups>
</workflow>
</extension>

```

This XML file specifies the custom task groups that are to appear in your custom **ACT Workflows** task group in the Workbench **Toolbox**. It performs the following actions:

- References the script `generic_material_transfer.py`.
- Defines a single task in the element **<tasks>**.
- Specifies that the callback **<onupdate>** invokes the function **producer_update**, which accesses a materials file.
- Defines the inputs and outputs in the element **<inputs>** and **<outputs>**. Note that an empty input and an output are defined. The attribute **outputtype** is set to **MatML31**, specifying the kind of data exposed and generated by this task.
- Defines two custom task groups in the element **<taskgroups>**.
 - The first custom task group, **GenericMaterialTransfer**, references the custom task **Material**, which transfers material data to an ANSYS-installed task group **Engineering Data** that has been added downstream.
 - The second custom task group, **GenericMaterialTransferComplete**, references the custom task **Material** and, by using the attribute **external**, also the ANSYS-installed task **Engineering Data**. The custom task transfers material data to the downstream ANSYS-installed task.

Defining Functions for Generic Material Transfer

The IronPython script `generic_material_transfer.py` defines the functions for passing the MatML-formatted material data to a downstream **Engineering Data** task group or task. This includes the method **update** (invoked by the callback **<onupdate>** in the XML file), which accesses the file `SampleMaterials.xml`.

```

def update(task):
    container = task.InternalObject
    extensionDir = ExtAPI.ExtensionManager.CurrentExtension.InstallDir
    matFilePath = System.IO.Path.Combine(extensionDir, "Sample_Materials.xml")
    matFileRef = None
    isRegistered = IsFileRegistered(FilePath=matFilePath)
    if isRegistered == True:
        matFileRef = GetRegisteredFile(matFilePath)
    else:
        matFileRef = RegisterFile(FilePath=matFilePath)
        AssociateFileWithContainer(matFileRef, container)
    outputRefs = container.GetOutputData()

```

Release 2020 R2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
<?xml version="1.0" encoding="UTF-8"?>
<EngineeringData version="16.1">
  <Notes />
  <Materials>
    <MatML_Doc>
      <Material>
        <BulkDetails>
          <Name>Sample Material</Name>
          <Description>Sample material from Driver</Description>
          <PropertyData property="pr0">
            <Data format="string">-</Data>
            <ParameterValue parameter="pa0" format="float">
              <Data>494.1474492,912.7972764,1172.453938,1941.495468,2803.754154,3869.063522,5245.395513,
10378.82012,18192.58268,28438.67868,57755.1982,94951.87682,135751.6191,178064.7612,216504.4272,261538.9311,
304701.5076,333300.2826,364061.2544,397079.5705,432533.1159,457543.8578,483751.5301</Data>
              <Qualifier name="Variable Type">Dependent,Dependent,Dependent,Dependent,Dependent,Dependent,Depend
            </ParameterValue>
            <ParameterValue parameter="pa1" format="float">
              <Data>0.1338,0.2675,0.3567,0.6242,0.8917,1.1592,1.4268,2.051,2.586,3.0318,3.7898,4.3694,4.8153,5.1
              <Qualifier name="Variable Type">Independent,Independent,Independent,Independent,Independent,Indepe
            </ParameterValue>
          </PropertyData>
          <PropertyData property="prDriver">
            <Data format="string">-</Data>
            <Qualifier name="Data Link Version">1</Qualifier>
            <Qualifier name="Model Type">Linear;Isotropic</Qualifier>
            <Qualifier name="Sample Property">Value</Qualifier>
          </PropertyData>
        </BulkDetails>
      </Material>
    </MatML_Doc>
  </Materials>
  <Metadata>
    <ParameterDetails id="pa0">
      <Name>Stress</Name>
      <Units>
        <Unit>
          <Name>Pa</Name>
        </Unit>
      </Units>
    </ParameterDetails>
    <ParameterDetails id="pa1">
      <Name>Strain</Name>
      <Units>
        <Unit>
          <Name>m</Name>
        </Unit>
        <Unit power="-1">
          <Name>m</Name>
        </Unit>
      </Units>
    </ParameterDetails>
    <PropertyDetails id="pr0">
      <Unitless />
      <Name>Sample Property</Name>
    </PropertyDetails>
    <PropertyDetails id="prDriver">
      <Unitless />
      <Name>Driver Link Details</Name>
    </PropertyDetails>
  </Metadata>
</EngineeringData>
```

```
</Metadata>  
</MatML_Doc>  
</Materials>  
<Loads />  
<BeamSections />  
</EngineeringData>
```

Workbench Wizards

You can use ACT to create project wizards for Workbench. The supplied extension **WizardDemos** contains a project wizard, multiple target product wizards, and a mixed wizard. The project wizard in this extension is fully described in [Wizard Creation](#) in the *ANSYS ACT Developer's Guide*.

Note:

You use the [Extension Manager](#) to install and load extensions and the [Wizards launcher](#) to start a wizard.

Tip:

Included in the supplied package [ACT Wizard Templates](#) is the folder **Template-Projects-*c*hematicWizard**. It contains an extension with a Workbench wizard that shows all the property capabilities for ACT and how to include reports and charts. For download information, see [Extension and Template Examples](#).

Appendix A. Component Input and Output Tables

The following tables list component inputs and outputs supported by ACT.

Table 1: 3D ROM

Taskgroup	Task	Input	Output
3D ROM	Design of Experiments (3D ROM)		DesignPointsDataTransfer
			DOEModel
			ParametricContext
	ROM Builder	DOEModel	RomBuilder
		ParametricContext	

Table 2: ACP (Post)

Taskgroup	Task	Input	Output
ACP (Post)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	

Taskgroup	Task	Input	Output
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Results	EngineeringData	
		MAPDLSolution	
		MechanicalSolution	
		SimulationGeneratedMesh	

Table 3: ACP (Pre)

Taskgroup	Task	Input	Output
ACP (Pre)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	

Taskgroup	Task	Input	Output
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	ACPSetupData	ACPSetupData
		EngineeringData	CompositeEngineeringData
		Geometry	EnhancedModelData
		SimulationEngineeringData	SimulationEngineeringData
		SimulationGeneratedMesh	SimulationModelGeneratedMesh
		SimulationGeneratedSolidMesh	SolidSectionData
		SimulationModelGeneratedMesh	

Table 4: Autodyn

Taskgroup	Task	Input	Output
Autodyn	Setup	AUTODYN_Remap	AutodynSetup
		MechanicalSetup	
		SimulationGeneratedMesh	
	Analysis		

Table 5: BladeGen

Taskgroup	Task	Input	Output
BladeGen	Blade Design		TurboGeometry
			VistaGeometry

Table 6: CFX

Taskgroup	Task	Input	Output
CFX	Setup	CFXMesh	CFXSetup
		FluentTGridMesh	SystemCouplingSetupData
		MechanicalSetup	
		SimulationGeneratedMesh	
	Solution	CFXSetup	CFXSolution
		CFXSolution	
	Results	CFXSolution	CFDAnalysis

Taskgroup	Task	Input	Output
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 7: CFX (Beta)

Taskgroup	Task	Input	Output
CFX (Beta)	Setup	CFXMesh	CFXSetup
		FluentTGridMesh	SystemCouplingSetupData
		MechanicalSetup	
		SimulationGeneratedMesh	
	Solution	CFXSetup	CFXSolution
		CFXSolution	

Table 8: Chemkin (Beta)

Taskgroup	Task	Input	Output
Chemkin (Beta)	Setup		ChemkinSetup
	Solution	ChemkinSetup	ChemkinData

Table 9: Coupled Field Static

Taskgroup	Task	Input	Output
Coupled Field Static	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	

Taskgroup	Task	Input	Output
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	AnsoftForceDataObject	MechanicalSetup
		CFXSolution	SimulationSetup
		EnhancedMechanicalModel	
		ExternalDataSetup	
		ExternalDataSetupForAqwa	
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults

Taskgroup	Task	Input	Output
			SimulationResults

Table 10: Coupled Field Transient

Taskgroup	Task	Input	Output
Coupled Field Transient	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOBJECT	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	AnsoftForceDataObject	MechanicalSetup
		CFXSolution	SimulationSetup
		EnhancedMechanicalModel	

Taskgroup	Task	Input	Output
		ExternalDataSetup	
		ExternalDataSetupForAqwa	
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
	Results		SimulationSolutionOutputProvider
		SimulationSolution	MechanicalResults
			SimulationResults

Table 11: Data Receive

Taskgroup	Task	Input	Output
Data Receive	Setup	ACPSetupData	DynardoTransfer
		CFXSolution	
		ExternalModelOutputProvider	
		FileProvider	
		FluentSolution	
		GeneralTransfer	
		Geometry	
		MAPDLResults	
		MechanicalSolution	
		MeshingMesh	
		MSExcelSetup	

Table 12: Data Receive (Beta)

Taskgroup	Task	Input	Output
Data Receive (Beta)	Setup		Geometry

Table 13: Data Send

Taskgroup	Task	Input	Output
Data Send	Setup	ACPSetupData	DynardoTransfer

Taskgroup	Task	Input	Output
		CFXSolution	
		ExternalModelOutputProvider	
		FileProvider	
		FluentSolution	
		GeneralTransfer	
		Geometry	
		MAPDLResults	
		MechanicalSolution	
		MeshingMesh	
		MSExcelsSetup	

Table 14: Designer Circuit

Taskgroup	Task	Input	Output
Designer Circuit	Setup		AnsoftCellInOutEntity
	Solution	AnsoftCellInOutEntity	

Table 15: Designer Circuit Netlist

Taskgroup	Task	Input	Output
Designer Circuit Netlist	Setup		AnsoftCellInOutEntity
	Solution	AnsoftCellInOutEntity	

Table 16: Direct Optimization

Taskgroup	Task	Input	Output
Direct Optimization	Optimization	DesignPointsDataTransfer	OptimizationModel

Table 17: Eigenvalue Buckling

Taskgroup	Task	Input	Output
Eigenvalue Buckling	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	

Taskgroup	Task	Input	Output
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	EnhancedMechanicalModel	MechanicalSetup
		GeneralTransfer	SimulationSetup
		MechanicalMesh	
		MechanicalModel	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	ExternalModelOutputProvider
			MechanicalResults

Taskgroup	Task	Input	Output
			SimulationResults

Table 18: EigenValue Buckling (Samcef)

Taskgroup	Task	Input	Output
Eigenvalue Buckling (Samcef)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	GeneralTransfer	MechanicalSetup
		MechanicalMesh	SimulationSetup
		MechanicalModel	

Taskgroup	Task	Input	Output
	Solution	SimulationSolutionDataInternal	
		GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
	Results	SimulationSolution	SimulationSolutionOutputProvider
			MechanicalResults
			SimulationResults

Table 19: Electric

Taskgroup	Task	Input	Output
Electric	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOBJECT	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	

Taskgroup	Task	Input	Output
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	GeneralTransfer	MechanicalSetup
		MechanicalMesh	SimulationSetup
		MechanicalModel	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 20: EnSight (Forte)

Taskgroup	Task	Input	Output
EnSight (Forte)	Results	ForteData	GeneralTransfer

Table 21: Engineering Data

Taskgroup	Task	Input	Output
Engineering Data	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	

Table 22: Explicit Dynamics

Taskgroup	Task	Input	Output
Explicit Dynamics	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	

Taskgroup	Task	Input	Output
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	EnhancedMechanicalModel	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		GeneralTransfer	
		MechanicalMesh	
		MechanicalModel	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults

Taskgroup	Task	Input	Output
			SimulationResults

Table 23: External Connection

Taskgroup	Task	Input	Output
External Connection	External Connection		ExternalConnectionProperties

Table 24: External Data

Taskgroup	Task	Input	Output
External Data	Setup		ExternalDataSetup
			ExternalMaterialFieldDataSetup
			ExternalTraceDataSetup

Table 25: External Model

Taskgroup	Task	Input	Output
External Model	Setup		ExternalModelOutputProvider

Table 26: Feedback Iterator

Taskgroup	Task	Input	Output
Feedback Iterator	Feedback Iterator	FeedbackIteratorSetup	Feedback IteratorEntity

Table 27: Fluent

Taskgroup	Task	Input	Output
Fluent	Setup	AIMFluentMeshOutputProvider	FluentSetup
		AIMFluentPartMeshFileOutputProvider	SystemCouplingSetupData
		AIMFluentPartMeshOutpuProvider	
		AIMFluentPhysicsDefinitionOutputProvider	
		AnsoftHeatLossDataObject	
		FluentCase	
		FluentImportable	
		FluentMesh	
		FluentTGridMesh	
		ICEData	
		ICESetupData	
		SimulationGeneratedMesh	
	Solution	FluentSetup	FluentSolution

Taskgroup	Task	Input	Output
		FluentSolution	

Table 28: Fluent (with CFD-Post) (Beta)

Taskgroup	Task	Input	Output
Fluent	Setup	AIMFluentMeshOutputProvider	FluentSetup
		AIMFluentPartMeshFileOutputProvider	SystemCouplingSetupData
		AIMFluentPartMeshOutputProvider	
		AIMFluentPhysicsDefinitionOutputProvider	
		AnsoftHeatLossDataObject	
		FluentCase	
		FluentImportable	
		FluentMesh	
		FluentTGridMesh	
		ICEData	
		ICESetupData	
		SimulationGeneratedMesh	
	Solution	FluentSetup	FluentSolution
		FluentSolution	
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 29: Fluent (with Fluent Meshing)

Taskgroup	Task	Input	Output
Fluent	Mesh	FluentMesh	FluentTGridMesh
		Geometry	
	Setup	AIMFluentMeshOutputProvider	FluentSetup
		AIMFluentPartMeshFileOutputProvider	SystemCouplingSetupData
		AIMFluentPartMeshOutputProvider	
		AIMFluentPhysicsDefinitionOutputProvider	
		AnsoftHeatLossDataObject	

Taskgroup	Task	Input	Output
		FluentCase	
		FluentImportable	
		FluentMesh	
		FluentTGridMesh	
		ICEData	
		ICESetupData	
		SimulationGeneratedMesh	
	Solution	FluentSetup	FluentSolution
		FluentSolution	

Table 30: Fluid Flow (CFX)

Taskgroup	Task	Input	Output
Fluid Flow (CFX)	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Mesh	Geometry	GeneratedMeshOutputForAQWAModelProvider
			MechanicalModel
		MeshingGeneratedMeshOutputProvider	MeshingGeneratedMeshOutputProvider
			MeshingMesh
			SimulationGeneratedMesh
	Setup	CFXMesh	CFXSetup
		FluentTGridMesh	SystemCouplingSetupData
		MechanicalSetup	
		SimulationGeneratedMesh	
	Solution	CFXSetup	CFXSolution
		CFXSolution	
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	

Taskgroup	Task	Input	Output
		NTIOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 31: Fluid Flow (Fluent)

Taskgroup	Task	Input	Output
Fluid Flow (Fluent)	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Mesh	Geometry	GeneratedMeshOutputForAQWAModelProvider
		MeshingGeneratedMeshOutputProvider	MechanicalModel
			MeshingGeneratedMeshOutputProvider
			MeshingMesh
	Setup	AIMFluentMeshOutputProvider	FluentSetup
		AIMFluentPartMeshFileOutputProvider	SystemCouplingSetupData
		AIMFluentPartMeshOutputProvider	
		AIMFluentPhysicsDefinitionOutputProvider	
		AnsoftHeatLossDataObject	
		FluentCase	
		FluentImportable	
		FluentMesh	
		FluentTGridMesh	
		ICEData	
		ICESetupData	
		SimulationGeneratedMesh	
	Solution	FluentSetup	FluentSolution
		FluentSolution	
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	

Taskgroup	Task	Input	Output
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 32: Fluid Flow (Polyflow)

Taskgroup	Task	Input	Output
Fluid Flow (Polyflow)	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Mesh	Geometry	GeneratedMeshOutputForAQWAModelProvider
			MechanicalModel
		MeshingGeneratedMeshOutputProvider	MeshingGeneratedMeshOutputProvider
			MeshingMesh
			SimulationGeneratedMesh
	Setup	FluentTGridMesh	PolyflowSetup
		Imported FLUENT Mesh File Type	
		PolyflowTransferMesh	
		SimulationGeneratedMesh	
	Solution	PolyflowSetup	ExternalDataSetup
		PolyflowSolution	PolyflowSolution
			PolyflowSolutionType
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	

Taskgroup	Task	Input	Output
		VistaTFSolution	

Table 33: Fluid Flow – Blow Molding (Polyflow)

Taskgroup	Task	Input	Output
Fluid Flow – Blow Molding (Polyflow)	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Mesh	Geometry	GeneratedMeshOutputForAQWAModelProvider
			MechanicalModel
		MeshingGeneratedMeshOutputProvider	MeshingGeneratedMeshOutputProvider
			MeshingMesh
	Setup	FluentTGridMesh	PolyflowSetup
		Imported FLUENT Mesh File Type	
		PolyflowTransferMesh	
		SimulationGeneratedMesh	
	Solution	PolyflowSetup	ExternalDataSetup
		PolyflowSolution	PolyflowSolution
			PolyflowSolutionType
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	

Taskgroup	Task	Input	Output
		VistaTFSolution	

Table 34: Fluid Flow – Extrusion (Polyflow)

Taskgroup	Task	Input	Output
Fluid Flow – Extrusion (Polyflow)	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDb	
		TurboGeometry	
	Mesh	Geometry	GeneratedMeshOutputForAQWAModelProvider
			MechanicalModel
		MeshingGeneratedMeshOutputProvider	MeshingGeneratedMeshOutputProvider
			MeshingMesh
	Setup	FluentTGridMesh	PolyflowSetup
		Imported FLUENT Mesh File Type	
		PolyflowTransferMesh	
		SimulationGeneratedMesh	
	Solution	PolyflowSetup	ExternalDataSetup
		PolyflowSolution	PolyflowSolution
			PolyflowSolutionType
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 35: Forte

Taskgroup	Task	Input	Output
Forte	Setup	Geometry	ForteData

Taskgroup	Task	Input	Output
		SimulationGeneratedMesh	
	Solution	ForteSetup	ForteData

Table 36: GRANTA MI

Taskgroup	Task	Input	Output
GRANTA MI	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	

Table 37: Geometry

Taskgroup	Task	Input	Output
Geometry	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	

Table 38: HFSS

Taskgroup	Task	Input	Output
HFSS	Geometry	AnsoftCADGeometryEntity	AnsoftCADObject
		AnsoftGeometryManagerDataObject	AnsoftCellInOutEntity
		Geometry	AnsoftGeometryManagerDataObject
	Setup	AnsoftCellInOutEntity	MatML31
			AnsoftCellInOutEntity
	Solution	AnsoftCellInOutEntity	FeedbackIteratorSetup
			AnsoftForceDataObject
			AnsoftHeatLossDataObject

Table 39: HFSS 3D Layout Design

Taskgroup	Task	Input	Output
HFSS 3D Layout Design	Setup		AnsoftCellInOutEntity

Taskgroup	Task	Input	Output
	Solution	AnsoftCellInOutEntity	

Table 40: HFSS-IE

Taskgroup	Task	Input	Output
HFSS-IE	Geometry	AnsoftCADGeometryEntity	AnsoftCADObject
		AnsoftGeometryManagerDataObject	AnsoftCellInOutEntity
		Geometry	AnsoftGeometryManagerDataObject
	Setup	AnsoftCellInOutEntity	MatML31
			AnsoftCellInOutEntity
	Solution	AnsoftCellInOutEntity	FeedbackIteratorSetup
			AnsoftHeatLossDataObject

Table 41: Harmonic Acoustics

Taskgroup	Task	Input	Output
Harmonic Acoustics	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDb	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDb
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	

Taskgroup	Task	Input	Output
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	ExternalMechanicalModel	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		GeneralTransfer	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 42: Harmonic Response

Taskgroup	Task	Input	Output
Harmonic Response	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData

Taskgroup	Task	Input	Output
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDb
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	AnsoftForceAndMomentDataObject	MechanicalSetup
		EnhancedMechanicalModel	SimulationSetup
		ExternalDataSetup	
		GeneralTransfer	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 43: Hydrodynamic Diffraction

Taskgroup	Task	Input	Output
Hydrodynamic Diffraction	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	

Taskgroup	Task	Input	Output
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	GeneratedMeshOutputForAQWAModelProvider	AqwaModel
		Geometry	
	Setup	AqwaModel	AqwaSetup
	Solution	AqwaSetup	AqwaSolution
			ExternalDataSetupForAqwa
	Results	AqwaSolution	AqwaResults

Table 44: Hydrodynamic Response

Taskgroup	Task	Input	Output
Hydrodynamic Response	Geometry	AnsoftCADObject	Geome
		FEMSetup	
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	GeneratedMeshOutputForAQWAModelProvider	AqwaM
		Geometry	
	Setup	AqwaModel	AqwaS
		AqwaSolution	
	Solution	AqwaSetup	AqwaS
			Externa
	Results	AqwaSolution	AqwaR

Table 45: IC Engine (Fluent)

Taskgroup	Task	Input	Output
IC Engine (Fluent)	ICE		ICEData
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	

Taskgroup	Task	Input	Output
	Mesh	TurboGeometry	
		Geometry	GeneratedMeshOutputForAQWAModelProvider
			MechanicalModel
		MeshingGeneratedMeshOutputProvider	MeshingGeneratedMeshOutputProvider
			MeshingMesh
			SimulationGeneratedMesh
	ICE Solver Setup	ForteSMGData	ICESetupData
		SimulationGeneralMesh	
		SimulationGeneratedMesh	
	Setup	AIMFluentMeshOutputProvider	FluentSetup
		AIMFluentPartMeshFileOutputProvider	SystemCouplingSetupData
		AIMFluentPartMeshOutputProvider	
		AIMFluentPhysicsDefinitionOutputProvider	
		AnsoftHeatLossDataObject	
		FluentCase	
		FluentImportable	
		FluentMesh	
		FluentTGridMesh	
		ICEData	
		ICESetupData	
		SimulationGeneratedMesh	
	Solution	FluentSetup	FluentSolution
		FluentSolution	
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 46: ICEM CFD

Taskgroup	Task	Input	Output
ICEM CFD	Model	FluentImportable	MeshingAssemblyTransferType
		Geometry	SimulationGeneratedMesh

Taskgroup	Task	Input	Output
		MechanicalMesh	
		MeshingMesh	
		Modeler	

Table 47: Icepak

Taskgroup	Task	Input	Output
Icepak	Setup	AnsoftHeatLossDataObject	IcePakSetup
		Geometry	
	Solution	IcePakSetup	IcePakResults

Table 48: Injection Molding Data (Beta)

Taskgroup	Task	Input	Output
Injection Molding Data (Beta)	Setup		ExternalDataSetup
			ExternalMaterialFieldDataSetup

Table 49: MOP Solver

Taskgroup	Task	Input	Output
MOP Solver	Setup	ACPSetupData	DynardoTransfer
		CFXSolution	
		ExternalModelOutputProvider	
		FileProvider	
		FluentSolution	
		GeneralTransfer	
		Geometry	
		MAPDLResults	
		MechanicalSolution	
		MeshingMesh	
		MSExcelsSetup	

Table 50: Magnetostatic

Taskgroup	Task	Input	Output
Magnetostatic	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	

Taskgroup	Task	Input	Output
		ICEDData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDb	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDb
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	GeneralTransfer	MechanicalSetup
		MechanicalMesh	SimulationSetup
		MechanicalModel	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 51: Material Designer

Taskgroup	Task	Input	Output
Material Designer	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material

Taskgroup	Task	Input	Output
		FEMSetup	
		MatML31	
	Material Designer	EngineeringData	MatML31

Table 52: Maxwell 2D

Taskgroup	Task	Input	Output
Maxwell 2D	Geometry	AnsoftCADGeometryEntity	AnsoftCADObject
		AnsoftGeometryManagerDataObject	AnsoftCellInOutEntity
		Geometry	AnsoftGeometryManagerDataObject
	Setup	AnsoftCellInOutEntity	MatML31
			AnsoftCellInOutEntity
			FeedbackIteratorSetup
	Solution	AnsoftCellInOutEntity	AnsoftForceAndMomentDataObject
			AnsoftForceDataObject
			AnsoftHeatLossDataObject

Table 53: Maxwell 3D

Taskgroup	Task	Input	Output
Maxwell 3D	Geometry	AnsoftCADGeometryEntity	AnsoftCADObject
		AnsoftGeometryManagerDataObject	AnsoftCellInOutEntity
		Geometry	AnsoftGeometryManagerDataObject
	Setup	AnsoftCellInOutEntity	MatML31
			AnsoftCellInOutEntity
			FeedbackIteratorSetup
	Solution	AnsoftCellInOutEntity	AnsoftForceAndMomentDataObject
			AnsoftForceDataObject
			AnsoftHeatLossDataObject

Table 54: Mechanical APDL

Taskgroup	Task	Input	Output
Mechanical APDL	Analysis	FEMSetup	
		Geometry	
		MAPDLCdb	
		MAPDLDatabase	
		MAPDLResults	
		MAPDLSolution	
		MechanicalSetup	

Taskgroup	Task	Input	Output
		MechanicalSolution	
		SimulationGeneratedMesh	
		SolidSectionData	

Table 55: Mechanical Model

Taskgroup	Task	Input	Output
Mechanical Model	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	

Taskgroup	Task	Input	Output
		TopologyOptimizationResultsCDB	

Table 56: Mesh

Taskgroup	Task	Input	Output
Mesh	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Mesh	Geometry	GeneratedMeshOutputForAQWAModelProvider
		MeshingGeneratedMeshOutputProvider	MechanicalModel
			MeshingGeneratedMeshOutputProvider
			MeshingMesh
			SimulationGeneratedMesh

Table 57: Microsoft Office Excel

Taskgroup	Task	Input	Output
Microsoft Office Excel	Analysis		MSExcelsSetup

Table 58: Modal

Taskgroup	Task	Input	Output
Modal	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh

Taskgroup	Task	Input	Output
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	EnhancedMechanicalModel	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		GeneralTransfer	
		MechanicalMesh	
		MechanicalModel	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	ExternalModelOutputProvider
			MechanicalResults
			SimulationResults

Table 59: Modal (ABAQUS)

Taskgroup	Task	Input	Output
Modal (ABAQUS)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	

Taskgroup	Task	Input	Output
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	GeneralTransfer	MechanicalSetup
		MechanicalMesh	SimulationSetup
		MechanicalModel	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults

Taskgroup	Task	Input	Output
			SimulationResults

Table 60: Modal (NASTRAN) (Beta)

Taskgroup	Task	Input	Output
Modal (NASTRAN) (Beta)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	GeneralTransfer	MechanicalSetup
		MechanicalMesh	SimulationSetup
		MechanicalModel	

Taskgroup	Task	Input	Output
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 61: Modal (Samcef)

Taskgroup	Task	Input	Output
Modal (Samcef)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	

Taskgroup	Task	Input	Output
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	GeneralTransfer	MechanicalSetup
		MechanicalMesh	SimulationSetup
		MechanicalModel	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 62: Modal Acoustics

Taskgroup	Task	Input	Output
Modal Acoustics	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	

Taskgroup	Task	Input	Output
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	EnhancedMechanicalModel	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		GeneralTransfer	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 63: Optimization

Taskgroup	Task	Input	Output
Optimization	Optimization System	DynardoTransferMOP	DynardoTransferOpti
		DynardoTransferOpti	
		DynardoTransferSensi	
		DynardoTransferValidator	

Table 64: Parameters Correlation

Taskgroup	Task	Input	Output
Parameters Correlation	Parameters Correlation	ResponseSurfaceDataTransfer	CorrelationModel

Taskgroup	Task	Input	Output
			DesignPointsDataTransfer

Table 65: Performance Map

Taskgroup	Task	Input	Output
Performance Map	Performance Map		

Table 66: Polyflow

Taskgroup	Task	Input	Output
Polyflow	Setup	FluentTGridMesh	PolyflowSetup
		Imported FLUENT Mesh File Type	
		PolyflowTransferMesh	
		SimulationGeneratedMesh	
	Solution	PolyflowSetup	ExternalDataSetup
		PolyflowSolution	PolyflowSolution
			PolyflowSolutionType

Table 67: Polyflow – Blow Molding

Taskgroup	Task	Input	Output
Polyflow – Blow Molding	Setup	FluentTGridMesh	PolyflowSetup
		Imported FLUENT Mesh File Type	
		PolyflowTransferMesh	
		SimulationGeneratedMesh	
	Solution	PolyflowSetup	ExternalDataSetup
		PolyflowSolution	PolyflowSolution
			PolyflowSolutionType

Table 68: Polyflow – Extrusion

Taskgroup	Task	Input	Output
Polyflow - Extrusion	Setup	FluentTGridMesh	PolyflowSetup
		Imported FLUENT Mesh File Type	
		PolyflowTransferMesh	
		SimulationGeneratedMesh	
	Solution	PolyflowSetup	ExternalDataSetup
		PolyflowSolution	PolyflowSolution

Taskgroup	Task	Input	Output
			PolyflowSolutionType

Table 69: Python (Beta)

Taskgroup	Task	Input	Output
Python	Setup	ACPSSetupData	DynardoTransfer
		CFXSolution	
		ExternalModelOutputProvider	
		FileProvider	
		FluentSolution	
		GeneralTransfer	
		Geometry	
		MAPDLResults	
		MechanicalSolution	
		MeshingMesh	
		MSExcelsSetup	

Table 70: Q3D 2D Extractor

Taskgroup	Task	Input	Output
Q3D 2D Extractor	Geometry	AnsoftCADGeometryEntity	AnsoftCADObject
		AnsoftGeometryManagerDataObject	AnsoftCellInOutEntity
		Geometry	AnsoftGeometryManagerDataObject
			MatML31
	Setup	AnsoftCellInOutEntity	AnsoftCellInOutEntity
			FeedbackIteratorSetup
	Solution	AnsoftCellInOutEntity	AnsoftForceDataObject
			AnsoftHeatLossDataObject

Table 71: Q3D Extractor

Taskgroup	Task	Input	Output
Q3D Extractor	Geometry	AnsoftCADGeometryEntity	AnsoftCADObject
		AnsoftGeometryManagerDataObject	AnsoftCellInOutEntity
		Geometry	AnsoftGeometryManagerDataObject
			MatML31
	Setup	AnsoftCellInOutEntity	AnsoftCellInOutEntity
			FeedbackIteratorSetup

Taskgroup	Task	Input	Output
	Solution	AnsoftCellInOutEntity	AnsoftHeatLossDataObject

Table 72: RMxpert

Taskgroup	Task	Input	Output
RMxpert	Setup		AnsoftCellInOutEntity
	Solution	AnsoftCellInOutEntity	

Table 73: Random Vibration

Taskgroup	Task	Input	Output
Random Vibration	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	

Taskgroup	Task	Input	Output
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	EnhancedMechanicalModel	MechanicalSetup
		GeneralTransfer	SimulationSetup
		MechanicalMesh	
		MechanicalModel	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 74: Response Spectrum

Taskgroup	Task	Input	Output
Response Spectrum	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedPMDB
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedMesh
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	

Taskgroup	Task	Input	Output
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	EnhancedMechanicalModel	MechanicalSetup
		GeneralTransfer	SimulationSetup
		MechanicalMesh	
		MechanicalModel	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 75: Response Surface

Taskgroup	Task	Input	Output
Response Surface	Design of Experiments		DesignPointsDataTransfer
			DOEModel
			ParametricContext
	Response Surface	DOEModel	DesignPointsDataTransfer
		ParametricContext	ResponseSurfaceDataTransfer
			ResponseSurfaceModel

Table 76: Response Surface Optimization

Taskgroup	Task	Input	Output
Response Surface Optimization	Design of Experiments		DesignPointsDataTransfer
			DOEModel
			ParametricContext
	Response Surface	DOEModel	DesignPointsDataTransfer
		ParametricContext	ResponseSurfaceDataTransfer
			ResponseSurfaceModel

Taskgroup	Task	Input	Output
	Optimization	ParametricContext	OptimizationModel
		ResponseSurfaceModel	

Table 77: Results

Taskgroup	Task	Input	Output
Results	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 78: Rigid Dynamics

Taskgroup	Task	Input	Output
Rigid Dynamics	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	

Taskgroup	Task	Input	Output
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	GeneralTransfer	MechanicalSetup
		MechanicalMesh	SimulationSetup
		MechanicalModel	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 79: Robustness

Taskgroup	Task	Input	Output
Robustness	Reliability System	DynardoTransferOpti	DynardoTransferReli
		DynardoTransferReli	
		DynardoTransferValidator	

Table 80: SPEOS

Taskgroup	Task	Input	Output
SPEOS	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	

Taskgroup	Task	Input	Output
	Simulation Task	Geometry	GeneralTransfer

Table 81: Sensitivity

Taskgroup	Task	Input	Output
Sensitivity	Sensitivity System	DynardoTransferSensi	DynardoTransferSensi
		DynardoTransferValidator	

Table 82: Sherlock (Post) (Beta)

Taskgroup	Task	Input	Output
Sherlock (Post)	Project		SherlockModelInternal
	Results	SherlockModelInternal	
		MechanicalSolution	

Table 83: Sherlock (Pre) (Beta)

Taskgroup	Task	Input	Output
Sherlock (Pre)	Project		SherlockModelInternal
	Setup	SherlockModelInternal	SherlockSetup
			SherlockGeometry

Table 84: Signal Processing

Taskgroup	Task	Input	Output
Signal Processing	Setup	ACPSSetupData	DynardoTransfer
		CFXSolution	
		ExternalModelOutputProvider	
		FileProvider	
		FluentSolution	
		GeneralTransfer	
		Geometry	
		MAPDLResults	
		MechanicalSolution	
		MeshingMesh	
		MSExcelSetup	

Table 85: Six Sigma Analysis

Taskgroup	Task	Input	Output
Six Sigma Analysis	Design of Experiments (SSA)		DesignPointsDataTransfer
			DOEModel
			ParametricContext

Taskgroup	Task	Input	Output
	Response Surface (SSA)	DOEModel	DesignPointsDataTransfer
		ParametricContext	ResponseSurfaceDataTransfer
			ResponseSurfaceModel
	Six Sigma Analysis	ParametricContext	SixSigmaModel
		ResponseSurfaceModel	

Table 86: Static Acoustics

Taskgroup	Task	Input	Output
Static Acoustics	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	

Taskgroup	Task	Input	Output
	Setup	TopologyOptimizationResultsCDB	
		AnsoftForceDataObject	MechanicalSetup
		CFXSolution	SimulationSetup
		EnhancedMechanicalModel	SystemCouplingSetupData
		ExternalDataSetup	
		ExternalDataSetupForAqwa	
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	ExternalModelOutputProvider
			MechanicalResults
			SimulationResults

Table 87: Static Mechanical

Taskgroup	Task	Input	Output
Static Mechanical	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData

Taskgroup	Task	Input	Output
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDb
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	AnsoftForceDataObject	MechanicalSetup
		CFXSolution	SimulationSetup
		EnhancedMechanicalModel	SystemCouplingSetupData
		ExternalDataSetup	
		ExternalDataSetupForAqwa	
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
		TopologySolutionLatticeData	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	ExternalModelOutputProvider
			MechanicalResults

Taskgroup	Task	Input	Output
			SimulationResults

Table 88: Static Structural

Taskgroup	Task	Input	Output
Static Structural	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	AnsoftForceDataObject	MechanicalSetup
		CFXSolution	SimulationSetup
		EnhancedMechanicalModel	SystemCouplingSetupData

Taskgroup	Task	Input	Output
		ExternalDataSetup	
		ExternalDataSetupForAqwa	
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
		TopologySolutionLatticeData	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
	Results	SimulationSolution	SimulationSolutionOutputProvider
			ExternalModelOutputProvider
			MechanicalResults
			SimulationResults

Table 89: Static Structural (ABAQUS)

Taskgroup	Task	Input	Output
Static Structural (ABAQUS)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh

Taskgroup	Task	Input	Output
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDb
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		Geometry	
		GeneralTransfer	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 90: Static Structural (Samcef)

Taskgroup	Task	Input	Output
Static Structural (Samcef)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	

Taskgroup	Task	Input	Output
		ICEDData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDb	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDb
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults

Taskgroup	Task	Input	Output
			SimulationResults

Table 91: Steady-State Thermal

Taskgroup	Task	Input	Output
Steady-State Thermal	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	AnsoftHeatLossDataObject	SimulationSetup
		CFXSolution	MechanicalSetup
		ExternalDataSetup	SystemCouplingSetupData

Taskgroup	Task	Input	Output
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
	Results	SimulationSolution	SimulationSolutionOutputProvider
			MechanicalResults
			SimulationResults

Table 92: Steady-State Thermal (ABAQUS)

Taskgroup	Task	Input	Output
Steady-State Thermal (ABAQUS)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	

Taskgroup	Task	Input	Output
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 93: Steady-State Thermal (MUSCADE)

Taskgroup	Task	Input	Output
Steady-State Thermal (MUSCADE)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	Engineering
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationC
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	

Taskgroup	Task	Input	Output
	Model	TurboGeometry	
		AIMGeometryMeshOutputProvider	Mechanical
		AIMMechanicalPhysicsDefinitionOutputProvider	Mechanical
		CompositeEngineeringData	SimulationE
		EngineeringData	SimulationC
		EnhancedModelData	SimulationC
		ExternalDataSetup	SimulationM
		ExternalMaterialFieldDataSetup	SimulationM
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	Mechanical
		ExternalDataSetup	SimulationS
		FluentSolution	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	SimulationSetup	Mechanical
			SimulationS
			SimulationS
			SimulationS
	Results	SimulationSolution	Mechanical
			SimulationF

Table 94: Steady-State Thermal (Samcef)

Taskgroup	Task	Input	Output
Steady-State Thermal	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material

Taskgroup	Task	Input	Output
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	

Taskgroup	Task	Input	Output
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 95: System Coupling

Taskgroup	Task	Input	Output
System Coupling	Setup	ExternalDataSetup	CouplingSetupProvider
		SystemCouplingSetupData	
	Solution	CouplingSetupProvider	

Table 96: Thermal-Electric

Taskgroup	Task	Input	Output
Thermal-Electric	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOBJECT	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		Geometry	
		GeneralTransfer	

Taskgroup	Task	Input	Output
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		FluentSolution	SystemCouplingSetupData
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 97: Throughflow

Taskgroup	Task	Input	Output
Throughflow	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Setup	Geometry	VistaTFSetup
		VistaGeometry	
		VistaTFPhysics	
	Solution	VistaTFSetup	VistaTFSolution
		VistaTFSolution	
	Results	CFXSolution	CFDAnalysis

Taskgroup	Task	Input	Output
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 98: Throughflow (BladeGen)

Taskgroup	Task	Input	Output
Throughflow (BladeGen)	Blade Design		TurboGeometry
			VistaGeometry
	Setup	Geometry	VistaTFSetup
		VistaGeometry	
		VistaTFPhysics	
	Solution	VistaTFSetup	VistaTFSolution
		VistaTFSolution	
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 99: Topology Optimization

Taskgroup	Task	Input	Output
Topology Optimization	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOObject	Geometry
		FEMSetup	SimulationGeneratedMesh

Taskgroup	Task	Input	Output
		Geometry	
		ICEDData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	GeneralTransfer	MechanicalSetup
		MechanicalMesh	SimulationSetup
		MechanicalModel	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
			TopologySolutionLatticeData
	Results	SimulationSolution	EngineeringData
			MechanicalResults
			SimulationResults

Taskgroup	Task	Input	Output
			TopologyOptimizationResultsCDB

Table 100: Transient Mechanical

Taskgroup	Task	Input	Output
Transient Mechanical	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	AnsoftForceDataObject	MechanicalSetup
		CFXSolution	SimulationSetup
		EnhancedMechanicalModel	SystemCouplingSetupData

Taskgroup	Task	Input	Output
		ExternalDataSetup	
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
	Results	SimulationSolution	SimulationSolutionOutputProvider
			ExternalModelOutputProvider
			MechanicalResults
			SimulationResults

Table 101: Transient Structural

Taskgroup	Task	Input	Output
Transient Structural	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	

Taskgroup	Task	Input	Output
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	AnsoftForceDataObject	MechanicalSetup
		CFXSolution	SimulationSetup
		EnhancedMechanicalModel	SystemCouplingSetupData
		ExternalDataSetup	
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
	Results		SimulationSolutionOutputProvider
		SimulationSolution	ExternalModelOutputProvider
			MechanicalResults
			SimulationResults

Table 102: Transient Structural (ABAQUS)

Taskgroup	Task	Input	Output
Transient Structural (ABAQUS)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh

Taskgroup	Task	Input	Output
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		Geometry	
		GeneralTransfer	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider

Taskgroup	Task	Input	Output
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 103: Transient Structural (Samcef)

Taskgroup	Task	Input	Output
Transient Structural (Samcef)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModalOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	MechanicalSetup
		ExternalDataSetup	SimulationSetup

Taskgroup	Task	Input	Output
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 104: Transient Thermal

Taskgroup	Task	Input	Output
Transient Thermal	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	

Taskgroup	Task	Input	Output
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	AnsoftHeatLossDataObject	MechanicalSetup
		CFXSolution	SimulationSetup
		ExternalDataSetup	SystemCouplingSetupData
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 105: Transient Thermal (ABAQUS)

Taskgroup	Task	Input	Output
Transient Thermal (ABAQUS)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADOObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	

Taskgroup	Task	Input	Output
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		FluentSolution	
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults

Taskgroup	Task	Input	Output
			SimulationResults

Table 106: Transient Thermal (Samcef)

Taskgroup	Task	Input	Output
Transient Thermal (Samcef)	Engineering Data	AIMEngineeringDataMaterialOutputProvider	EngineeringData
		ExternalModelOutputProvider	Material
		FEMSetup	
		MatML31	
	Geometry	AnsoftCADObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Model	AIMGeometryMeshOutputProvider	MechanicalMesh
		AIMMechanicalPhysicsDefinitionOutputProvider	MechanicalModel
		CompositeEngineeringData	SimulationEngineeringData
		EngineeringData	SimulationGeneratedMesh
		EnhancedModelData	SimulationGeneratedSolidMesh
		ExternalDataSetup	SimulationModelGeneratedMesh
		ExternalMaterialFieldDataSetup	SimulationModelGeneratedPMDB
		ExternalModelOutputProvider	
		ExternalTraceDataSetup	
		GeneralTransfer	
		Geometry	
		MeshingAssemblyTransferType	
		Modeler	
		SherlockSetup	
		SimulationEngineeringData	
		SimulationModelGeneratedMesh	
		SimulationSolutionOutputProvider	
		SolidSectionData	
		TopologyOptimizationResultsCDB	
	Setup	CFXSolution	MechanicalSetup
		ExternalDataSetup	SimulationSetup
		FluentSolution	

Taskgroup	Task	Input	Output
		GeneralTransfer	
		IcePakResults	
		MechanicalMesh	
		MechanicalModel	
		MechanicalSolution	
		SimulationSolutionDataInternal	
	Solution	GeneralTransfer	MechanicalSolution
		SimulationSetup	SimulationSolution
			SimulationSolutionDataInternal
			SimulationSolutionOutputProvider
	Results	SimulationSolution	MechanicalResults
			SimulationResults

Table 107: Turbo Setup

Taskgroup	Task	Input	Output
Turbo Setup	Turbo Setup		

Table 108: TurboGrid

Taskgroup	Task	Input	Output
TurboGrid	Turbo Mesh	Geometry	CFXMesh
		TurboGeometry	FluentImportable
			TurboMesh

Table 109: Turbomachinery Fluid Flow

Taskgroup	Task	Input	Output
Turbomachinery Fluid Flow	Turbo Mesh	Geometry	CFXMesh
		TurboGeometry	FluentImportable
			TurboMesh
	Setup	CFXMesh	CFXSetup
		FluentTGridMesh	SystemCouplingSetup Data
		MechanicalSetup	
		SimulationGeneratedMesh	
	Solution	CFXSetup	CFXSolution
		CFXSolution	
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	

Taskgroup	Task	Input	Output
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 110: Turbomachinery Fluid Flow (BladeEditor) (Beta)

Taskgroup	Task	Input	Output
Turbomachinery Fluid Flow (BladeEditor) (Beta)	Geometry	AnsoftCADOObject	Geometry
		FEMSetup	SimulationGeneratedMesh
		Geometry	
		ICEData	
		MechanicalResults	
		SherlockGeometry	
		SimulationModelGeneratedPMDB	
		TurboGeometry	
	Turbo Mesh	Geometry	CFXMesh
		TurboGeometry	FluentImportable
			TurboMesh
	Setup	CFXMesh	CFXSetup
		FluentTGridMesh	SystemCouplingSetupData
		MechanicalSetup	
		SimulationGeneratedMesh	
	Solution	CFXSetup	CFXSolution
		CFXSolution	
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	

Taskgroup	Task	Input	Output
		VistaTFSolution	

Table 111: Turbomachinery Fluid Flow (BladeGen) (Beta)

Taskgroup	Task	Input	Output
Turbomachinery Fluid Flow (BladeGen) (Beta)	Blade Design		TurboGeometry
			VistaGeometry
	Turbo Mesh	Geometry	CFXMesh
		TurboGeometry	FluentImportable
			TurboMesh
	Setup	CFXMesh	CFXSetup
		FluentTGridMesh	SystemCouplingSetupData
		MechanicalSetup	
		SimulationGeneratedMesh	
	Solution	CFXSetup	CFXSolution
		CFXSolution	
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 112: Twin Builder

Taskgroup	Task	Input	Output
Twin Builde	Setup	MechanicalSetup	AnsoftCellInOutEntity
	Solution	AnsoftCellInOutEntity	

Table 113: Vista AFD

Taskgroup	Task	Input	Output
Vista AFD	Meanline		VistaAFDMeanlineProvider
	Design	VistaAFDMeanlineProvider	VistaAFDDesignProvider

Taskgroup	Task	Input	Output
	Analysis	VistaAFDDesignProvider	

Table 114: Vista CCD

Taskgroup	Task	Input	Output
Vista CCD	Blade Design		VistaCCDBladeDesignProvider

Table 115: Vista CCD (with CCM)

Taskgroup	Task	Input	Output
Vista CCD (with CCM)	Blade Design		VistaCCDBladeDesignProvider
	Performance Map	VistaCCDBladeDesignProvider	

Table 116: Vista CPD

Taskgroup	Task	Input	Output
Vista CPD	Blade Design		

Table 117: Vista RTD

Taskgroup	Task	Input	Output
Vista RTD	Blade Design		

Table 118: Vista RTD (Beta)

Taskgroup	Task	Input	Output
Vista RTD (Beta)	Blade Design		VistaGeometry
			VistaTFPhysics

Table 119: Vista TF

Taskgroup	Task	Input	Output
Vista TF	Setup	Geometry	VistaTFSetup
		VistaGeometry	
		VistaTFPhysics	
	Solution	VistaTFSetup	VistaTFSolution
		VistaTFSolution	
	Results	CFXSolution	CFDAnalysis
		FluentSolution	
		ForteData	
		ForteSolution	

Taskgroup	Task	Input	Output
		ICEData	
		IcePakResults	
		MechanicalSolution	
		NTIOutput	
		PolyflowSolutionType	
		VistaTFSolution	

Table 120: OptiSLang Pre (Beta)

Taskgroup	Task	Input	Output
OptiSLang Pre (Beta)		ETK	CFXMesh
			CFXSetup
			CFXSolution
			EngineeringData
			ExternalData
			FluentCase
			FluentMesh
			FluentSetup
			FluentSolution
			Geometry
			MAPDLCdb
			MAPDLDatabase
			MAPDLResults
			MatML31
			MechanicalMesh
			MechanicalModel
			MechanicalSetup
			MechanicalSolution
			MeshingMesh

Appendix B. ANSYS-Installed System Component Template and Display Names

Table 121: 3D ROM

Component Name	Component Display Name
DXDOECeIlForRomBuilderTemplate	Design of Experiments (3D ROM)
DXRomBuilderCeIlTemplate	ROM Builder

Table 122: ACP (Post)

Component Name	Component Display Name
EngDataCeIlTemplate	Engineering Data
GeometryCeIlTemplate	Geometry
SimulationModelCeIlTemplate	Model
ACPResultsCeIlTemplate	Results

Table 123: ACP (Pre)

Component Name	Component Display Name
EngDataCeIlTemplate	Engineering Data
GeometryCeIlTemplate	Geometry
SimulationModelCeIlTemplate	Model
ACPSetupCeIlTemplate	Setup

Table 124: Autodyn

Component Name	Component Display Name
AUTODYN_Solution	Setup
AUTODYN_Results	Analysis

Table 125: BladeGen

Component Name	Component Display Name
TSGeometryTemplate	Blade Design

Table 126: CFX

Component Name	Component Display Name
CFXPhysicsTemplate	Setup
CFXResultsTemplate	Solution

Component Name	Component Display Name
CFDPostTemplate	Results

Table 127: CFX (Beta)

Component Name	Component Display Name
CFXPhysicsTemplate	Setup
CFXResultsTemplate	Solution

Table 128: Chemkin (Beta)

Component Name	Component Display Name
ChemkinSetupCellTemplate	Setup
ChemkinSolutionCellTemplate	Solution

Table 129: Coupled Field Static

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_CustomizableDesignAssessmentANSYS	Setup
SimulationSolutionCellTemplate_CustomizableDesignAssessmentANSYS	Solution
SimulationResultsCellTemplate_CustomizableDesignAssessmentANSYS	Results

Table 130: Coupled Field Transient

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_CustomizableDesignAssessmentANSYS	Setup
SimulationSolutionCellTemplate_CustomizableDesignAssessmentANSYS	Solution

Component Name	Component Display Name
SimulationResultsCellTemplate_CustomizableDesignAssessmentANSYS	Results

Table 131: Data Receive

Component Name	Component Display Name
PDMReceiveTemplate	Setup

Table 132: Data Receive (Beta)

Component Name	Component Display Name
PDMReceiveTemplate	Setup

Table 133: Data Send

Component Name	Component Display Name
PDMSendTemplate	Setup

Table 134: Designer Circuit

Component Name	Component Display Name
DesignerCircuitSetup	Setup
DesignerCircuitSolution	Solution

Table 135: Designer Circuit Netlist

Component Name	Component Display Name
NexximNetlistSetup	Setup
NexximNetlistSolution	Solution

Table 136: Direct Optimization

Component Name	Component Display Name
DXDirectOptimizationCellTemplate	Optimization

Table 137: Eigenvalue Buckling

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralBucklingANSYS	Setup
SimulationSolutionCellTemplate_StructuralBucklingANSYS	Solution

Component Name	Component Display Name
SimulationResultsCellTemplate_StructuralBucklingANSYS	Results

Table 138: Eigenvalue Buckling (Samcef)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralBucklingSamcef	Setup
SimulationSolutionCellTemplate_StructuralBucklingSamcef	Solution
SimulationResultsCellTemplate_StructuralBucklingSamcef	Results

Table 139: Electric

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_ElectricStaticANSYS	Setup
SimulationSolutionCellTemplate_ElectricStaticANSYS	Solution
SimulationResultsCellTemplate_ElectricStaticANSYS	Results

Table 140: EnSight (Forte)

Component Name	Component Display Name
ResultsForteTemplate	Results

Table 141: Engineering Data

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data

Table 142: Explicit Dynamics

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralExplicitAUTODYN	Setup
SimulationSolutionCellTemplate_StructuralExplicitAUTODYN	Solution

Component Name	Component Display Name
SimulationResultsCellTemplate_StructuralExplicitAUTODYN	Results

Table 143: External Connection

Component Name	Component Display Name
ExternalConnectionTemplate	External Connection

Table 144: External Data

Component Name	Component Display Name
ExternalLoadSetupCellTemplate	Setup

Table 145: External Model

Component Name	Component Display Name
ExternalModelSetupCellTemplate	Setup

Table 146: Feedback Iterator

Component Name	Component Display Name
FeedbackIteratorComponentTemplate	Feedback Iterator

Table 147: Fluent

Component Name	Component Display Name
FluentSetupCellTemplate	Setup
FluentResultsCellTemplate	Solution

Table 148: Fluent (with CFD-Post) (Beta)

Component Name	Component Display Name
FluentSetupCellTemplate	Setup
FluentResultsCellTemplate	Solution
CFDPostTemplate	Results

Table 149: Fluent (with Fluent Meshing)

Component Name	Component Display Name
FluentTGridCellTemplate	Mesh
FluentSetupCellTemplate	Setup
FluentResultsCellTemplate	Solution

Table 150: Fluid Flow (CFX)

Component Name	Component Display Name
GeometryCellTemplate	Geometry
SimulationMeshingModelCellTemplate	Mesh

Component Name	Component Display Name
CFXPhysicsTemplate	Setup
CFXResultsTemplate	Solution
CFDPostTemplate	Results

Table 151: Fluid Flow (Fluent)

Component Name	Component Display Name
GeometryCellTemplate	Geometry
SimulationMeshingModelCellTemplate	Mesh
FluentSetupCellTemplate	Setup
FluentResultsCellTemplate	Solution
CFDPostTemplate	Results

Table 152: Fluid Flow (Polyflow)

Component Name	Component Display Name
GeometryCellTemplate	Geometry
SimulationMeshingModelCellTemplate	Mesh
PolyflowSetupCellTemplate	Setup
PolyflowSolveCellTemplate	Solution
CFDPostTemplate	Results

Table 153: Fluid Flow - Blow Molding (Polyflow)

Component Name	Component Display Name
GeometryCellTemplate	Geometry
SimulationMeshingModelCellTemplate	Mesh
PolyflowSetupCellTemplate	Setup
PolyflowSolveCellTemplate	Solution
CFDPostTemplate	Results

Table 154: Fluid Flow - Extrusion (Polyflow)

Component Name	Component Display Name
GeometryCellTemplate	Geometry
SimulationMeshingModelCellTemplate	Mesh
PolyflowSetupCellTemplate	Setup
PolyflowSolveCellTemplate	Solution
CFDPostTemplate	Results

Table 155: Forte

Component Name	Component Display Name
ForteSetupCellTemplate	Setup

Component Name	Component Display Name
ForteSolutionCellTemplate	Solution

Table 156: GRANTA MI

Component Name	Component Display Name
GrantaComponentTemplate	MI Materials Gateway
EngDataCellTemplate	Engineering Data

Table 157: Geometry

Component Name	Component Display Name
GeometryCellTemplate	Geometry

Table 158: HFSS

Component Name	Component Display Name
HFSSGeometry	Geometry
HFSSSetup	Setup
HFSSSolution	Solution

Table 159: HFSS 3D Layout Design

Component Name	Component Display Name
DesignerEMSetup	Setup
DesignerEMSolution	Solution

Table 160: HFSS-IE

Component Name	Component Display Name
HFSS-IEGeometry	Geometry
HFSS-IESetup	Setup
HFSS-IESolution	Solution

Table 161: Harmonic Acoustics

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_MechanicalAcousticsMultiphysicsHarmonicANSYS	Setup
SimulationSolutionCellTemplate_MechanicalAcousticsMultiphysicsHarmonicANSYS	Solution

Component Name	Component Display Name
SimulationResultsCellTemplate_MechanicalAcousticsMultiphysicsHarmonicANSYS	ANSYS

Table 162: Harmonic Response

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_MechanicalAcousticsMultiphysicsHarmonicANSYS	Setup
SimulationSolutionCellTemplate_MechanicalAcousticsMultiphysicsHarmonicANSYS	Solution
SimulationResultsCellTemplate_MechanicalAcousticsMultiphysicsHarmonicANSYS	Results

Table 163: Hydrodynamic Diffraction

Component Name	Component Display Name
GeometryCellTemplate	Geometry
AQWAModelCellTemplate	Model
AQWASetupHDCellTemplate	Setup
AQWAAAnalysisCellTemplate	Solution
AQWAResultsCellTemplate	Results

Table 164: Hydrodynamic Response

Component Name	Component Display Name
GeometryCellTemplate	Geometry
AQWAModelCellTemplate	Model
AQWASetupHDCellTemplate	Setup
AQWAAAnalysisCellTemplate	Solution
AQWAResultsCellTemplate	Results

Table 165: IC Engine (Fluent)

Component Name	Component Display Name
ICEComponentTemplate	ICE
GeometryCellTemplate	Geometry
SimulationMeshingModelCellTemplate	Mesh
ICESetupComponentTemplate	ICE Solver Setup
FluentSetupCellTemplate	Setup
FluentResultsCellTemplate	Solution

Component Name	Component Display Name
CFDPostTemplate	Results

Table 166: ICEM CFD

Component Name	Component Display Name
ICEMCFD	Model

Table 167: Icepak

Component Name	Component Display Name
IcePakSetupCellTemplate	Setup
IcePakSolutionCellTemplate	Solution

Table 168: Injection Molding Data (Beta)

Component Name	Component Display Name
InjectionMoldingDataSetupTemplate	Setup

Table 169: MOP Solver

Component Name	Component Display Name
MopsolverTemplate	Setup

Table 170: Magnetostatic

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_ElectromagneticsMagnetostaticsANSYS	Setup
SimulationSolutionCellTemplate_ElectromagneticsMagnetostaticsANSYS	Solution
SimulationResultsCellTemplate_ElectromagneticsMagnetostaticsANSYS	Results

Table 171: Material Designer

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
MaterialDesignerCellTemplate	Material Designer

Table 172: Maxwell 2D

Component Name	Component Display Name
Maxwell2DGeometry	Geometry
Maxwell2DSetup	Setup

Component Name	Component Display Name
Maxwell2DSolution	Solution

Table 173: Maxwell 3D

Component Name	Component Display Name
Maxwell3DGeometry	Geometry
Maxwell3DSetup	Setup
Maxwell3DSolution	Solution

Table 174: Mechanical APDL

Component Name	Component Display Name
ANSYSSetupCellTemplate	Analysis

Table 175: Mechanical Model

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model

Table 176: Mesh

Component Name	Component Display Name
GeometryCellTemplate	Geometry
SimulationMeshingModelCellTemplate	Mesh

Table 177: Microsoft Office Excel

Component Name	Component Display Name
MSEExcelComponentTemplate	Analysis

Table 178: Modal

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralModalANSYS	Setup
SimulationSolutionCellTemplate_StructuralModalANSYS	Solution
SimulationResultsCellTemplate_StructuralModalANSYS	Results

Table 179: Modal (ABAQUS)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data

Component Name	Component Display Name
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralModalABAQUS	Setup
SimulationSolutionCellTemplate_StructuralModalABAQUS	Solution
SimulationResultsCellTemplate_StructuralModalABAQUS	Results

Table 180: Modal (NASTRAN) (Beta)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralModalNASTRAN	Setup
SimulationSolutionCellTemplate_StructuralModalNASTRAN	Solution
SimulationResultsCellTemplate_StructuralModalNASTRAN	Results

Table 181: Modal (Samcef)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralModalSamcef	Setup
SimulationSolutionCellTemplate_StructuralModalSamcef	Solution
SimulationResultsCellTemplate_StructuralModalSamcef	Results

Table 182: Modal Acoustics

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_MechanicalAcousticsMultiphysicsModalANSYS	Setup
SimulationSolutionCellTemplate_MechanicalAcousticsMultiphysicsModalANSYS	Solution

Component Name	Component Display Name
SimulationResultsCellTemplate_MechanicalAcousticsMultiphysicsModalANSYS	Results

Table 183: Optimization

Component Name	Component Display Name
OptimizationTaskTemplate	Optimization System

Table 184: Parameters Correlation

Component Name	Component Display Name
DXCorrelationCellTemplate	Parameters Correlation

Table 185: Performance Map

Component Name	Component Display Name
PerfMapTemplate	Performance Map

Table 186: Polyflow

Component Name	Component Display Name
PolyflowSetupCellTemplate	Setup
PolyflowSolveCellTemplate	Solution

Table 187: Polyflow - Blow Molding

Component Name	Component Display Name
PolyflowBlowModlingSetupCellTemplate	Setup
PolyflowBlowModlingSolveCellTemplate	Solution

Table 188: Polyflow - Extrusion

Component Name	Component Display Name
PolyflowExtrusionSetupCellTemplate	Setup
PolyflowExtrusionSolveCellTemplate	Solution

Table 189: Python (Beta)

Component Name	Component Display Name
Python2Template	Setup

Table 190: Q3D 2D Extractor

Component Name	Component Display Name
Q3D2DExtractorGeometry	Geometry
Q3D2DExtractorSetup	Setup

Component Name	Component Display Name
Q3D2DExtractorSolution	Solution

Table 191: Q3D Extractor

Component Name	Component Display Name
Q3DExtractorGeometry	Geometry
Q3DExtractorSetup	Setup
Q3DExtractorSolution	Solution

Table 192: RMxport

Component Name	Component Display Name
RMxpprtSetup	Setup
RMxpprtSolution	Solution

Table 193: Random Vibration

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralRandomVibrationANSYS	Setup
SimulationSolutionCellTemplate_StructuralRandomVibrationANSYS	Solution
SimulationResultsCellTemplate_StructuralRandomVibrationANSYS	Results

Table 194: Response Spectrum

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralResponseSpectrumANSYS	Setup
SimulationSolutionCellTemplate_StructuralResponseSpectrumANSYS	Solution
SimulationResultsCellTemplate_StructuralResponseSpectrumANSYS	Results

Table 195: Response Surface

Component Name	Component Display Name
DXDOECeIlTemplate	Design of Experiments
DXResponseSurfaceCellTemplate	Response Surface

Table 196: Response Surface Optimization

Component Name	Component Display Name
DXDOECeIlTemplate	Design of Experiments

Component Name	Component Display Name
DXResponseSurfaceCellTemplate	Response Surface
DXOptimizationCellTemplate_GDO	Optimization

Table 197: Results

Component Name	Component Display Name
CFDPostTemplate	Results

Table 198: Rigid Dynamics

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralTransientRBD	Setup
SimulationSolutionCellTemplate_StructuralTransientRBD	Solution
SimulationResultsCellTemplate_StructuralTransientRBD	Results

Table 199: Robustness

Component Name	Component Display Name
ReliabilityTaskTemplate	Reliability System

Table 200: SPEOS

Component Name	Component Display Name
GeometryCellTemplate	Geometry
SimulationTaskTemplate	Simulation Task

Table 201: Sensitivity

Component Name	Component Display Name
SensitivityTaskTemplate	Sensitivity System

Table 202: Sherlock (Post) (Beta)

Component Name	Component Display Name
SherlockModelCellTemplate	Project
SherlockResultsCellTemplate	Results

Table 203: Sherlock (Pre) (Beta)

Component Name	Component Display Name
SherlockModelCellTemplate	Project

Component Name	Component Display Name
SherlockSetupCellTemplate	Setup

Table 204: Signal Processing

Component Name	Component Display Name
ETKCompleteTemplate	Setup

Table 205: Six Sigma Analysis

Component Name	Component Display Name
DXDOECellForSixSigmaTemplate	Design of Experiments (SSA)
DXResponseSurfaceCellForSixSigmaTemplate	Response Surface (SSA)
DXSixSigmaCellTemplate	Six Sigma Analysis

Table 206: Static Acoustics

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_MechanicalAcousticsMultiphysicsStaticANSYS	Setup
SimulationSolutionCellTemplate_MechanicalAcousticsMultiphysicsStaticANSYS	Solution
SimulationResultsCellTemplate_MechanicalAcousticsMultiphysicsStaticANSYS	Results

Table 207: Static Mechanical

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_MechanicalMultiphysicsStaticANSYS	Setup
SimulationSolutionCellTemplate_MechanicalMultiphysicsStaticANSYS	Solution
SimulationResultsCellTemplate_MechanicalMultiphysicsStaticANSYS	Results

Table 208: Static Structural

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralStaticANSYS	Setup
SimulationSolutionCellTemplate_StructuralStaticANSYS	Solution

Component Name	Component Display Name
SimulationResultsCellTemplate_StructuralStaticANSYS	Results

Table 209: Static Structural (ABAQUS)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralStaticABAQUS	Setup
SimulationSolutionCellTemplate_StructuralStaticABAQUS	Solution
SimulationResultsCellTemplate_StructuralStaticABAQUS	Results

Table 210: Static Structural (Samcef)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralStaticSamcef	Setup
SimulationSolutionCellTemplate_StructuralStaticSamcef	Solution
SimulationResultsCellTemplate_StructuralStaticSamcef	Results

Table 211: Steady-State Thermal

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_ThermalSteadyStateANSYS	Setup
SimulationSolutionCellTemplate_ThermalSteadyStateANSYS	Solution
SimulationResultsCellTemplate_ThermalSteadyStateANSYS	Results

Table 212: Steady-State Thermal (ABAQUS)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_ThermalSteadyStateABAQUS	Setup
SimulationSolutionCellTemplate_ThermalSteadyStateABAQUS	Solution

Component Name	Component Display Name
SimulationResultsCellTemplate_ThermalSteadyStateABAQUS	Results

Table 213: Steady-State Thermal (MUSCADE)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_ThermalSteadyStateMUSCADE	Setup
SimulationSolutionCellTemplate_ThermalSteadyStateMUSCADE	Solution
SimulationResultsCellTemplate_ThermalSteadyStateMUSCADE	Results

Table 214: Steady-State Thermal (Samcef)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_ThermalSteadyStateSamcef	Setup
SimulationSolutionCellTemplate_ThermalSteadyStateSamcef	Solution
SimulationResultsCellTemplate_ThermalSteadyStateSamcef	Results

Table 215: System Coupling

Component Name	Component Display Name
SystemCouplingSetupCellTemplate	Setup
SystemCouplingSolutionCellTemplate	Solution

Table 216: Thermal-Electric

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_ThermalElectromagneticsStaticANSYS	Setup
SimulationSolutionCellTemplate_ThermalElectromagneticsStaticANSYS	Solution
SimulationResultsCellTemplate_ThermalElectromagneticsStaticANSYS	Results

Table 217: Throughflow

Component Name	Component Display Name
GeometryCellTemplate	Geometry
VistaTFSetupTemplate	Setup
VistaTFSolutionTemplate	Solution

Component Name	Component Display Name
CFDPostTemplate	Results

Table 218: Throughflow (BladeGen)

Component Name	Component Display Name
TSGeometryTemplate	Blade Design
VistaTFSetupTemplate	Setup
VistaTFSolutionTemplate	Solution
CFDPostTemplate	Results

Table 219: Topology Optimization

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralShapeOptimizationANSYS	Setup
SimulationSolutionCellTemplate_StructuralShapeOptimizationANSYS	Solution
SimulationResultsCellTemplate_StructuralShapeOptimizationANSYS	Results

Table 220: Transient Mechanical

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_MechanicalMultiphysicsTransientANSYS	Setup
SimulationSolutionCellTemplate_MechanicalMultiphysicsTransientANSYS	Solution
SimulationResultsCellTemplate_MechanicalMultiphysicsTransientANSYS	Results

Table 221: Transient Structural

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralTransientANSYS	Setup
SimulationSolutionCellTemplate_StructuralTransientANSYS	Solution
SimulationResultsCellTemplate_StructuralTransientANSYS	Results

Table 222: Transient Structural (ABAQUS)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data

Component Name	Component Display Name
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralTransientABAQUS	Setup
SimulationSolutionCellTemplate_StructuralTransientABAQUS	Solution
SimulationResultsCellTemplate_StructuralTransientABAQUS	Results

Table 223: Transient Structural (Samcef)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_StructuralTransientSamcef	Setup
SimulationSolutionCellTemplate_StructuralTransientSamcef	Solution
SimulationResultsCellTemplate_StructuralTransientSamcef	Results

Table 224: Transient Thermal

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_ThermalTransientANSYS	Setup
SimulationSolutionCellTemplate_ThermalTransientANSYS	Solution
SimulationResultsCellTemplate_ThermalTransientANSYS	Results

Table 225: Transient Thermal (ABAQUS)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model
SimulationSetupCellTemplate_ThermalTransientABAQUS	Setup
SimulationSolutionCellTemplate_ThermalTransientABAQUS	Solution
SimulationResultsCellTemplate_ThermalTransientABAQUS	Results

Table 226: Transient Thermal (Samcef)

Component Name	Component Display Name
EngDataCellTemplate	Engineering Data
GeometryCellTemplate	Geometry
SimulationModelCellTemplate	Model

Component Name	Component Display Name
SimulationSetupCellTemplate_ThermalTransientSamcef	Setup
SimulationSolutionCellTemplate_ThermalTransientSamcef	Solution
SimulationResultsCellTemplate_ThermalTransientSamcef	Results

Table 227: Turbo Setup

Component Name	Component Display Name
TSSetupTemplate	Turbo Setup

Table 228: TurboGrid

Component Name	Component Display Name
TSMeshTemplate	Turbo Mesh

Table 229: Turbomachinery Fluid Flow

Component Name	Component Display Name
TSMeshTemplate	Turbo Mesh
CFXPhysicsTemplate	Setup
CFXResultsTemplate	Solution
CFDPostTemplate	Results

Table 230: Turbomachinery Fluid Flow (BladeEditor) (Beta)

Component Name	Component Display Name
GeometryCellTemplate	Geometry
TSMeshTemplate	Turbo Mesh
CFXPhysicsTemplate	Setup
CFXResultsTemplate	Solution
CFDPostTemplate	Results

Table 231: Turbomachinery Fluid Flow (BladeGen) (Beta)

Component Name	Component Display Name
TSGeometryTemplate	Blade Design
TSMTSMeshTemplate	Turbo Mesh
CFXPhysicsTemplate	Setup
CFXResultsTemplate	Solution
CFDPostTemplate	Results

Table 232: Twin Builder

Component Name	Component Display Name
SimplorerSetup	Setup

Component Name	Component Display Name
SimplorerSolution	Solution

Table 233: Vista AFD

Component Name	Component Display Name
TSVistaAFDMeanlineTemplate	Meanline
TSVistaAFDDesignTemplate	Design
TSVistaAFDAnalysisTemplate	Analysis

Table 234: Vista CCD

Component Name	Component Display Name
TSVistaCCDTemplate	Blade Design

Table 235: Vista CCD (with CCM)

Component Name	Component Display Name
TSVistaCCDTemplate	Blade Design
TSVistaCCMTemplate	Performance Map

Table 236: Vista CPD

Component Name	Component Display Name
TSVistaCPDTemplate	Blade Design

Table 237: Vista RTD

Component Name	Component Display Name
TSVistaRTDTemplate	Blade Design

Table 238: Vista RTD (Beta)

Component Name	Component Display Name
TSVistaRTDTemplate	Blade Design

Table 239: Vista TF

Component Name	Component Display Name
VistaTFSetupTemplate	Setup
VistaTFSolutionTemplate	Solution
CFDPostTemplate	Results

Table 240: OptiSLang Pre (Beta)

Component Name	Component Display Name

Appendix C. Data Transfer Types

Table 241: Data Transfer Types and Properties

Transfer Type	Property
ACPSData	ACPFileReference
	ACPPreFileReference
AIMEngineeringDataMaterialOutputProvider	
AIMFluentMeshOutputProvider	
AIMFluentPartMeshFileOutputProvider	
AIMFluentPartMeshOutputProvider	
AIMFluentPhysicsDefinitionOutputProvider	
AIMGeometryMeshOutputProvider	
AIMMechanicalPhysicsDefinitionOutputProvider	
AnsoftCADGeometryEntity	
AnsoftCADObject	
AnsoftCellInOutEntity	
AnsoftForceAndMomentDataObject	AnsoftTransferXMLString
	AnsoftProjectResultsFolderAtCurrentDP
AnsoftForceDataObject	AnsoftTransferXMLString
	AnsoftProjectResultsFolderAtCurrentDP
AnsoftGeometryManagerDataObject	
AnsoftHeatLossDataObject	AnsoftTransferXMLString
	AnsoftProjectResultsFolderAtCurrentDP
AqwaModel	
AqwaResults	
AqwaSetup	
AqwaSolution	
AUTODYN_Remap	
AutodynSetup	
CFDAnalysis	PostStateFile
ChemkinData	TransferFile
ChemkinSetup	TransferFile
CFXMesh	FileName
	PreFileType
CFXSetup	CFXSolverInputFile

Transfer Type	Property
	MAPDLSolverInputFile
CFXSolution	MResLoadOption
	CFXResultsFile
	AuxiliaryFiles
	MAPDLResultsFile
CompositeEngineeringData	TransferFile
CorrelationModel	
CouplingSetupProvider	TransferFile
DesignPointsDataTransfer	
DOEModel	
DynardoTransfer	
DynardoTransferMOP	
DynardoTransferOpti	
DynardoTransferReli	
DynardoTransferSensi	
DynardoTransferValidator	
EngineeringData	TransferFile
EnhancedMechanicalModel	
EnhancedModelData	
ETK	
ExternalDataSetup	TransferFile
ExternalDataSetupForAqwa	
ExternalMaterialFieldDataSetup	
ExternalModelOutputProvider	TransferFile
	InputFiles
ExternalTraceDataSetup	
FeedbackIteratorEntity	
FeedbackIteratorSetup	
FileProvider	
FEMMesh	ACMOFile
FEMSetup	FEModelerFile
	ANSYSInputFile
	ParasolidFile
	FiniteElementModelMaterials
	AuxiliaryFiles
FluentCase	MeshFile
	TransferFile
FluentImportable	MeshFile

Transfer Type	Property
	FileType
	Dimension
FluentMesh	TransferFile
FluentSetup	CaseFile
	ModelInfoFile
FluentSolution	CaseFile
	DataFile
FluentTGridMesh	TransferFile
ForteData	
ForteSetup	
ForteSetupData	DataFile
ForteSMGData	
ForteSolution	
ForteSolutionData	
GeneralTransfer	
GeneratedMeshOutputForAQWAModelProvider	PMDBFile
	ACMOFile
	Mechdb
Geometry	GeometryFilePath
	PlugInName
ICEData	
IcePakResults	
IcePakSetup	
ICESetupData	
Imported FLUENT Mesh File Type	MeshFile
MAPDLCdb	TransferFile
	AuxiliaryFiles
MAPDLDatabase	TransferFile
	AuxiliaryFiles
MAPDLResults	AuxiliaryFiles
MAPDLSolution	TransferFile
	AuxiliaryFiles
Material	
MatML31	TransferFile
MechanicalMesh	TransferFile
MechanicalModel	File
	EdaFile
MechanicalResults	

Transfer Type	Property
MechanicalSetup	TransferFile
MechanicalSolution	
MeshingAssemblyTransferType	TransferFile
MeshingGeneratedMeshOutputProvider	PMDBFile
	ACMOFile
	Mechdb
MeshingMesh	TransferFile
Modeler	
MSExcelsSetup	
NTIOutput	
OptimizationModel	
ParametricContext	
PolyflowSetup	
PolyflowSolution	
PolyflowSolutionType	DataFile
	PubFile
	GeneratedFiles
PolyflowTransferMesh	TransferFile
ResponseSurfaceDataTransfer	
ResponseSurfaceModel	
SherlockGeometry	GeometryFile
SherlockModelInternal	
SherlockSetup	AnalysisType
	BoardInfoFile
	GeometryFile
	JournalFile
	ProviderVersion
	PythonScriptFile
SimulationEngineeringData	TransferFiles
SimulationGeneralMesh	TransferFile
SimulationGeneratedMesh	TransferFile
SimulationGeneratedSolidMesh	
SimulationModelGeneratedMesh	TransferFile
SimulationModelGeneratedPMDB	TransferFile
SimulationResults	
SimulationSetup	
SimulationSolution	
SimulationSolutionDataInternal	

Transfer Type	Property
SimulationSolutionOutputProvider	
SixSigmaModel	
SolidSectionData	TransferFile
	AuxiliaryFiles
	CompositeSectionFiles
SystemCouplingSetupData	
TopologyOptimizationResultsCDB	TransferFiles
TopologySolutionLatticeData	
TurboGeometry	INFFilename
	GeometryFilename
TurboMesh	FileName
VistaAFDDesignProvider	TransferData
VistaAFDMeanlineProvider	TransferData
VistaCCDBladeDesignProvider	TransferData
VistaGeometry	GeoData
	TransferData
VistaTFPhysics	TransferData
VistaTFSetup	ControlFilename
	GeoFilename
	AeroFilename
VistaTFSetup	CorrelationsFilename
VistaTFSolution	ResultsFile
	RestartFile

Appendix D. Addin Data Types and Data Transfer Formats

The following table lists the data types and data transfer formats that are supported for each addin.

Note:

Where listed, a value of "n/a" indicates that you should use an empty format string.

Addin	Data Type	Format
ACP	ACPSetupData	n/a
AEDT (HFSS, Maxwell, Q3D, RMprt)	AnsoftCADGeometryEntity	n/a
	AnsoftCADObject	n/a
	AnsoftCellInOutEntity	n/a
	AnsoftForceAndMomentDataObject	n/a
	AnsoftForceDataObject	n/a
	AnsoftGeometryDataObject	n/a
	AnsoftGeometryManagerDataObject	n/a
	AnsoftHeatLossDataObject	n/a
AIM	AIMEngineeringDataMaterialOutputProvider	n/a
	AIMFluentPartMeshFileOutputProvider	n/a
	AIMFluentPartMeshOutputProvider	n/a
	AIMFluentPhysicsDefinitionOutputProvider	n/a
	AIMMechanicalPhysicsDefinitionOutputProvider	n/a
ANSYS	MAPDLCdb	n/a
	MAPDLDatabase	n/a
	MAPDLResults	n/a
	MAPDLSolution	n/a
AQWA	AqwaModel	transfer not supported
	AqwaResults	transfer not supported
	AqwaSetup	transfer not supported
	AqwaSolution	transfer not supported
AUTODYN	AutodynAnalysis	transfer not supported
	AutodynSetup	transfer not supported
CFX	CFXSetup	n/a
	CFXSolution	n/a
Chemkin	ChemkinData	n/a

Addin	Data Type	Format
	ChemkinSetup	n/a
DesignModeler	Geometry	n/a
DesignXplorer	DesignPointsDataTransfer	n/a
	DOEModel	n/a
	ParametricContext	n/a
	RomBuilder	n/a
Engineering Data	EngineeringData	materials
External Connection	ExternalConnectionProperties	n/a
External Load	ExternalDataSetup	ExternalDataSetup
	ExternalDataSetupForAqwa	ExternalDataSetupForAqwa
	ExternalMaterialFieldDataSetup	ExternalMaterialFieldDataSetup
	ExternalModelOutputProvider	ExternalModelOutputProvider
	ExternalTraceDataSetup	ExternalTraceDataSetup
ExternalModelCoupling	EnhancedModelData	EnhancedModelData
	SolidSectionData	n/a
FEModeler	FEMMesh	ACMOFile
	FEMSetup	FacetedGeometry
		ParasolidGeometry
		InputFile
		FiniteElementModelMaterial
Fluent	FluentSetup	transfer not supported
	FluentSolution	n/a
	FluentTGridMesh	n/a
Forte	ForteData	transfer not supported
	ForteSetup	ForteSetup
	ForteSetupData	n/a
	ForteSMGData	transfer not supported
	ForteSolution	transfer not supported
	ForteSolutionData	n/a
ICEMCFD	MeshingAssemblyTransferType	FluentMesh
		Imported FLUENT Mesh FileType
		CMDDBMesh
		POLYFLOWMesh
		AnsysMesh
		ICEMCFDMesh
ICEngine	ICEData	ICEData
	ICESetupData	n/a

Addin	Data Type	Format
IcePak	IcePakSetup	transfer not supported
	IcePakResults	n/a
Meshing	GeneratedMeshForAQWAModelProvider	n/a
	MeshingGeneratedMeshOutputProvider	n/a
	MeshingMesh	CMDBMesh
		FluentMesh
		POLYFLOWMesh
MSExcel	MSExcelSetup	transfer not supported
Multiphysics Coupling	CouplingSetupProvider	CouplingSetupProvider
	SystemCouplingSetupData	n/a
NTI	NTIOutput	n/a
OptiSLang	DynardoTransfer	transfer not supported
	DynardoTransferMOP	transfer not supported
	DynardoTransferOpti	transfer not supported
	DynardoTransferReli	transfer not supported
	DynardoTransferSensi	transfer not supported
	DynardoTransferValidator	transfer not supported
	ETK	transfer not supported
	FileProvider	transfer not supported
Polyflow	PolyflowSetup	transfer not supported
	PolyflowSolution	transfer not supported
	PolyflowSolutionType	n/a
Sherlock	SherlockGeometry	n/a
	SherlockModelInternal	n/a
	SherlockSetup	n/a
Simulation	GeneralTransfer	n/a
	MechanicalMesh	CMDBMesh
		FluentMesh
		POLYFLOWMesh
	MechanicalModel	transfer not supported
	MechanicalResults	n/a
	MechanicalSetup	n/a
	MechanicalSolution	n/a
	SimulationEngineeringData	SimulationEngineeringData
	SimulationGeneratedMesh	AnsysisMesh
		CMDBMesh
		FluentMesh

Addin	Data Type	Format
		(Imported Fluent Mesh File Type)
		POLYFLOWMesh
	SimulationGeneratedSolidMesh	AnsysMesh
		CMDBMesh
		FluentMesh
		(Imported Fluent Mesh File Type)
		POLYFLOWMesh
	SimulationModelGeneratedMesh	n/a
	SimulationModelGeneratedPMDB	n/a
	SimulationResults	n/a
	SimulationSetup	n/a
	SimulationSolution	transfer not supported
	SimulationSolutionDataInternal	transfer not supported
TurboSystem	SimulationSolutionOutputProvider	n/a
	TopologySolutionLatticeData	n/a
	CFXMesh	n/a
	FluentImportable	n/a
	TurboMesh	transfer not supported
	VistaAFDDesignProvider	n/a
	VistaAFDMeanlineProvider	n/a
	VistaATFPhysics	n/a
VistaTF	VistaCCDBladeDesignProvider	n/a
	VistaGeometry	n/a
	VistaTFSetup	n/a
	VistaTFSolution	n/a

Appendix E. ANSYS-Installed Custom Workflow Support

The following components can consume generic data transfer provided by ACT custom tasks. For an explanation of generic data transfer, see [Defining Task-Level Data Transfer \(p. 63\)](#).

Table 242: Task Groups and Tasks

Task Groups	Tasks
ACP (Post)	Model
ACP (Pre)	Model
Coupled Field Static	Model
	Setup
	Solution
Coupled Field Transient	Model
	Setup
	Solution
Data Receive	Setup
Data Send	Setup
Eigenvalue Buckling	Model
	Setup
	Solution
Eigenvalue Buckling (Samcef)	Model
	Setup
	Solution
Electric	Model
	Setup
	Solution
Explicit Dynamics	Model
	Setup
	Solution
Harmonic Acoustics	Model
	Setup
	Solution
Harmonic Response	Model
	Setup
	Solution
MOP Solver	Setup

Task Groups	Tasks
Magnetostatic	Model
	Setup
	Solution
Mechanical Model	Model
Modal	Model
	Setup
	Solution
Modal (ABAQUS)	Model
	Setup
	Solution
Modal (NASTRAN) (Beta)	Model
	Setup
	Solution
Modal (Samcef)	Model
	Setup
	Solution
Modal Acoustics	Model
	Setup
	Solution
Python	Setup
Random Vibration	Model
	Setup
	Solution
Response Spectrum	Model
	Setup
	Solution
Rigid Dynamics	Model
	Setup
	Solution
Response Spectrum	Model
	Setup
	Solution
Signal Processing	Setup
Static Acoustics	Model
	Setup
	Solution
Static Structural	Model
	Setup

Task Groups	Tasks
	Solution
Static Structural (ABAQUS)	Model
	Setup
	Solution
Static Structural (Samcef)	Model
	Setup
	Solution
Steady-State Thermal	Model
	Setup
	Solution
Steady-State Thermal (ABAQUS)	Model
	Setup
	Solution
Steady-State Thermal (Samcef)	Model
	Setup
	Solution
Thermal-Electric	Model
	Setup
	Solution
Topological Optimization	Model
	Setup
	Solution
Transient Structural	Model
	Setup
	Solution
Transient Structural (ABAQUS)	Model
	Setup
	Solution
Transient Structural (Samcef)	Model
	Setup
	Solution
Transient Thermal	Model
	Setup
	Solution
Transient Thermal (ABAQUS)	Model
	Setup
	Solution
Transient Thermal (Samcef)	Model

Task Groups	Tasks
	Setup
	Solution