coreform.mesh

Table of contents

Source directory: `interop/abaqus/cf_app/coreform/mesh`

Directory tree

- `mesh/`

1 model.py

Module path: `coreform.mesh.model`

Available API members

Name	Kind	Description
`IGAMesh.spec`	Attribute	Concrete portable Mesh specification to wrap.
`IGAMesh.mesh_type`	Attribute	Return the type of the wrapped Mesh specification.
`IGAMesh.immersed_rectilinear`	Class method	Create an immersed rectilinear Mesh from a spec or spec fields.
`IGAMesh.body_fit`	Class method	Create a body-fit Mesh from a spec or spec fields.
`IGAMesh.from_spec`	Class method	Wrap an existing portable Mesh specification.
`IGAMesh.to_spec`	Instance method	Return the wrapped portable Mesh specification.
`IGAMesh.copy_with`	Instance method	Return a validated Mesh copy with selected fields replaced.
`IGAMesh.is_immersed_rectilinear`	Instance method	Return whether this Mesh wraps an immersed rectilinear specification.
`IGAMesh.is_body_fit`	Instance method	Return whether this Mesh wraps a body-fit specification.
`IGAMesh`	Class	Wrap one portable Mesh specification.

1.1 `IGAMesh`

Object path: `coreform.mesh.model.IGAMesh`

Wrap one portable Mesh specification.

Call syntax

```
result = IGAMesh(spec=spec)
```

Signatures

```
IGAMesh(spec: MeshSpec)
```

Parameters

Name	Type	Required	Default	Description
`spec`	`MeshSpec`	Yes	—	Concrete portable Mesh specification to wrap.

Example

```
>>> mesh = IGAMesh.immersed_rectilinear(
...     degree=2,
...     continuity=1,
...     sizing=ElementSizeSizing((0.5, 0.5, 0.5)),
... )
```

1.1.a Attributes

Name	Type	Value	Description
`spec`	`MeshSpec`	—	Concrete portable Mesh specification to wrap.
`mesh_type`	`MeshType`	—	Return the type of the wrapped Mesh specification.

1.1.b `immersed_rectilinear`

Defined on: `coreform.mesh.model.IGAMesh.immersed_rectilinear`

Description: Create an immersed rectilinear Mesh from a spec or spec fields.

Call syntax

```
result = IGAMesh.immersed_rectilinear(spec)
result = IGAMesh.immersed_rectilinear(sizing=sizing)
```

Returns

Type	Description
`IGAMesh`	Create an immersed rectilinear Mesh from a spec or spec fields.

Signatures

```
immersed_rectilinear(spec: ImmersedRectilinearMeshSpec, /) -> IGAMesh
immersed_rectilinear(degree: int = 1, continuity: int = 0, bc_penalty_value: Optional[float] = None, tessellation_size: Optional[float] = None, sizing: Optional[RectilinearSizing] = None, element_layout: Optional[Tuple[str, str, str]] = None, small_cell_volume_ratio: float = 0.2, padding: Optional[Tuple[int, int, int]] = None, frame: Optional[Frame] = None, extents: Optional[Tuple[Float3, Float3]] = None) -> IGAMesh
```

Parameters

For `immersed_rectilinear(spec: ImmersedRectilinearMeshSpec, /) -> IGAMesh`:

Name	Type	Required	Default	Description
`spec`	`ImmersedRectilinearMeshSpec`	Yes	—	Complete portable specification.

For `immersed_rectilinear(degree: int = 1, continuity: int = 0, bc_penalty_value: Optional[float] = None, tessellation_size: Optional[float] = None, sizing: Optional[RectilinearSizing] = None, element_layout: Optional[Tuple[str, str, str]] = None, small_cell_volume_ratio: float = 0.2, padding: Optional[Tuple[int, int, int]] = None, frame: Optional[Frame] = None, extents: Optional[Tuple[Float3, Float3]] = None) -> IGAMesh`:

Name	Type	Required	Default	Description
`degree`	`int`	No	`1`	Polynomial degree of the Mesh basis.
`continuity`	`int`	No	`0`	Continuity of the Mesh basis; it must be less than `degree`.
`bc_penalty_value`	`Optional[float]`	No	`None`	Positive penalty used to

Name	Type	Required	Default	Description
				impose essential boundary conditions.
`tessellation_size`	`Optional[float]`	No	`None`	Positive target tessellation size, or `None` for the default.
`sizing`	`Optional[RectilinearSizing]`	Yes	—	Required element-size or interval-count sizing specification.
`element_layout`	`Optional[Tuple[str, str, str]]`	No	`None`	Per-axis `edge_centered` or `node_centered` layout choices.
`small_cell_volume_ratio`	`float`	No	`0.2`	Small-cell threshold in the half-open interval `[0, 0.5)`.
`padding`	`Optional[Tuple[int, int, int]]`	No	`None`	Nonnegative cell padding along each frame axis.
`frame`	`Optional[Frame]`	No	`None`	Explicit local frame; provide it together with `extents`.
`extents`	`Optional[Tuple[Float3, Float3]]`	No	`None`	Increasing minimum and maximum corners in frame coordinates.

1.1.c `body_fit`

Defined on: `coreform.mesh.model.IGAMesh.body_fit`

Description: Create a body-fit Mesh from a spec or spec fields.

Call syntax

```
result = IGAMesh.body_fit(spec)
result = IGAMesh.body_fit()
```

Returns

Type	Description
`IGAMesh`	Create a body-fit Mesh from a spec or spec fields.

Signatures

```
body_fit(spec: BodyFitMeshSpec, /) -> IGAMesh
body_fit(degree: int = 1, continuity: int = 0, bc_penalty_value: Optional[float]
= None, tessellation_size: Optional[float] = None) -> IGAMesh
```

Parameters

For `body_fit(spec: BodyFitMeshSpec, /) -> IGAMesh`:

Name	Type	Required	Default	Description
`spec`	`BodyFitMeshSpec`	Yes	—	Complete portable specification.

For `body_fit(degree: int = 1, continuity: int = 0, bc_penalty_value: Optional[float] = None, tessellation_size: Optional[float] = None) -> IGAMesh`:

Name	Type	Required	Default	Description
`degree`	`int`	No	`1`	Polynomial degree of the Mesh basis.
`continuity`	`int`	No	`0`	Continuity of the Mesh basis; it must be less than `degree`.

Name	Type	Required	Default	Description
`bc_penalty_value`	`Optional[float]`	No	`None`	Positive penalty used to impose essential boundary conditions.
`tessellation_size`	`Optional[float]`	No	`None`	Positive target tessellation size, or `None` for the default.

1.1.d `from_spec`

Defined on: `coreform.mesh.model.IGAMesh.from_spec`

Description: Wrap an existing portable Mesh specification.

Call syntax

```
result = IGAMesh.from_spec(spec=spec)
```

Returns

Type	Description
`IGAMesh`	Wrap an existing portable Mesh specification.

Signatures

```
from_spec(spec: MeshSpec) -> 'IGAMesh'
```

Parameters

Name	Type	Required	Default	Description
`spec`	`MeshSpec`	Yes	—	Concrete portable Mesh specification to wrap.

1.1.e `to_spec`

Defined on: `coreform.mesh.model.IGAMesh.to_spec`

Description: Return the wrapped portable Mesh specification.

Call syntax

```
result = mesh.to_spec()
```

Returns

Type	Description
`MeshSpec`	Return the wrapped portable Mesh specification.

Signatures

```
to_spec() -> MeshSpec
```

Parameters

None.

1.1.f `copy_with`

Defined on: `coreform.mesh.model.IGAMesh.copy_with`

Description: Return a validated Mesh copy with selected fields replaced.

Call syntax

```
result = mesh.copy_with(...)
```

Returns

Type	Description
`IGAMesh`	Return a validated Mesh copy with selected fields replaced.

Signatures

```
copy_with(*, degree: int = ..., continuity: int = ..., bc_penalty_value:
Optional[float] = ..., tessellation_size: Optional[float] = ..., sizing:
Optional[RectilinearSizing] = ..., element_layout: Optional[Tuple[str, str,
str]] = ..., small_cell_volume_ratio: float = ..., padding: Optional[Tuple[int,
int, int]] = ..., frame: Optional[Frame] = ..., extents: Optional[Tuple[Float3,
Float3]] = ...) -> IGAMesh
copy_with(*, degree: int = ..., continuity: int = ..., bc_penalty_value:
Optional[float] = ..., tessellation_size: Optional[float] = ...) -> IGAMesh
```

Parameters

For `copy_with(*, degree: int = ..., continuity: int = ..., bc_penalty_value: Optional[float] = ..., tessellation_size: Optional[float] = ..., sizing: Optional[RectilinearSizing] = ..., element_layout: Optional[Tuple[str, str, str]] = ..., small_cell_volume_ratio: float = ..., padding: Optional[Tuple[int, int, int]] = ..., frame: Optional[Frame] = ..., extents: Optional[Tuple[Float3, Float3]] = ...) -> IGAMesh`:

Name	Applies to	Type	Required	Default	Description
`degree`	All	`int`	No	Current value	Polynomial degree of the Mesh basis.
`continuity`	All	`int`	No	Current value	Continuity of the Mesh basis; it must be less than `degree`.
`bc_penalty_value`	All	`Optional[float]`	No	Current value	Positive penalty used to impose essential boundary conditions.
`tessellation_size`	All	`Optional[float]`	No	Current value	Positive target tessellation size, or `None` for the default.
`sizing`	`ImmersedRectilinearMesh`	`Optional[RectilinearSizing]`	No	Current value	Required element-size or interval-count sizing specification.
`element_layout`	`ImmersedRectilinearMesh`	`Optional[Tuple[str, str, str]]`	No	Current value	Per-axis `edge_centered` or `node_centered`

Name	Applies to	Type	Required	Default	Description
					layout choices.
`small_cell_volume_ratio`	`ImmersedRectilinearMeshSpec`	`float`	No	Current value	Small-cell threshold in the half-open interval `[0, 0.5)`.
`padding`	`ImmersedRectilinearMeshSpec`	`int | tuple[int, int, int, int, int, int]`	No	Current value	Nonnegative cell padding along each frame axis.
`frame`	`ImmersedRectilinearMeshSpec`	`Frame`	No	Current value	Explicit local frame; provide it together with `extents`.
`extents`	`ImmersedRectilinearMeshSpec`	`tuple[Float3, Float3]`	No	Current value	Increasing minimum and maximum corners in frame coordinates.

For `copy_with(*, degree: int = ..., continuity: int = ..., bc_penalty_value: Optional[float] = ..., tessellation_size: Optional[float] = ...) -> IGAMesh`:

Name	Applies to	Type	Required	Default	Description
`degree`	All	`int`	No	Current value	Polynomial degree of the Mesh basis.
`continuity`	All	`int`	No	Current value	Continuity of the Mesh ba-

Name	Applies to	Type	Required	Default	Description
					sis; it must be less than `degree`.
`bc_penalty_value`	All	`Optional[float]`	No	Current value	Positive penalty used to impose essential boundary conditions.
`tessellation_size`	All	`Optional[float]`	No	Current value	Positive target tessellation size, or `None` for the default.

Parameter rules

Rule	Requirement
Applicability	`sizing` applies only to `ImmersedRectilinearMeshSpec`.
Applicability	`element_layout` applies only to `ImmersedRectilinearMeshSpec`.
Applicability	`small_cell_volume_ratio` applies only to `ImmersedRectilinearMeshSpec`.
Applicability	`padding` applies only to `ImmersedRectilinearMeshSpec`.
Applicability	`frame` applies only to `ImmersedRectilinearMeshSpec`.
Applicability	`extents` applies only to `ImmersedRectilinearMeshSpec`.

1.1.g `is_immersed_rectilinear`

Defined on: `coreform.mesh.model.IGAMesh.is_immersed_rectilinear`

Description: Return whether this Mesh wraps an immersed rectilinear specification.

Call syntax

```
result = mesh.is_immersed_rectilinear()
```

Returns

Type	Description
`bool`	Return whether this Mesh wraps an immersed rectilinear specification.

Signatures

```
is_immersed_rectilinear() -> bool
```

Parameters

None.

1.1.h `is_body_fit`

Defined on: `coreform.mesh.model.IGAMesh.is_body_fit`

Description: Return whether this Mesh wraps a body-fit specification.

Call syntax

```
result = mesh.is_body_fit()
```

Returns

Type	Description
`bool`	Return whether this Mesh wraps a body-fit specification.

Signatures

```
is_body_fit() -> bool
```

Parameters

None.

2 specs.py

Module path: `coreform.mesh.specs`

Available API members

Name	Kind	Description
`MeshType.IMMERSED_RECTILINEAR`	Attribute	Immersed rectilinear Mesh specification.
`MeshType.BODY_FIT`	Attribute	Body-fit Mesh specification.
`ElementSizeSizing.schemaVersion`	Attribute	Version of the serialized specification schema.
`ElementSizeSizing.element_size`	Attribute	Positive target element size along each frame axis.
`ElementSizeSizing.sizing_type`	Attribute	Serialized discriminator for this sizing form.
`ElementIntervalCountByAxisSizing.schemaVersion`	Attribute	Version of the serialized specification schema.
`ElementIntervalCountByAxisSizing.element_interval_count_by_axis`	Attribute	Positive element-interval count along each frame axis.
`ElementIntervalCountByAxisSizing.sizing_type`	Attribute	Serialized discriminator for this sizing form.
`ElementIntervalCountSizing.schemaVersion`	Attribute	Version of the serialized specification schema.
`ElementIntervalCountSizing.element_interval_count`	Attribute	Positive element-interval count for the three frame axes.
`ElementIntervalCountSizing.sizing_type`	Attribute	Serialized discriminator for this sizing form.
`RectilinearSizing`	Attribute	Any supported rectilinear Mesh sizing specification.
`Frame.schemaVersion`	Attribute	Version of the serialized specification schema.
`Frame.origin`	Attribute	Origin of the local frame in global coordinates.
`Frame.basis`	Attribute	Right-handed orthonormal basis vectors stored as rows.

Name	Kind	Description
`ImmersedRectilinearMeshSpec.schema_version`	Attribute	Version of the serialized specification schema.
`ImmersedRectilinearMeshSpec.degree`	Attribute	Polynomial degree of the Mesh basis.
`ImmersedRectilinearMeshSpec.continuity`	Attribute	Continuity of the Mesh basis; it must be less than `degree`.
`ImmersedRectilinearMeshSpec.penalty_value`	Attribute	Positive penalty used to impose essential boundary conditions.
`ImmersedRectilinearMeshSpec.tessellation_size`	Attribute	Positive target tessellation size, or `None` for the default.
`ImmersedRectilinearMeshSpec.sizing`	Attribute	Required element-size or interval-count sizing specification.
`ImmersedRectilinearMeshSpec.axis_layout`	Attribute	Per-axis `edge_centered` or `node_centered` layout choices.
`ImmersedRectilinearMeshSpec.small_cell_volume_ratio`	Attribute	Small-cell threshold in the half-open interval `[0, 0.5)`.
`ImmersedRectilinearMeshSpec.padding`	Attribute	Nonnegative cell padding along each frame axis.
`ImmersedRectilinearMeshSpec.frame`	Attribute	Explicit local frame; provide it together with `extents`.
`ImmersedRectilinearMeshSpec.extents`	Attribute	Increasing minimum and maximum corners in frame coordinates.
`ImmersedRectilinearMeshSpec.mesh_type`	Attribute	Persistent Mesh discriminator for this specification.
`BodyFitMeshSpec.schema_version`	Attribute	Version of the serialized specification schema.
`BodyFitMeshSpec.degree`	Attribute	Polynomial degree of the Mesh basis.

Name	Kind	Description
`BodyFitMeshSpec.continuity`	Attribute	Continuity of the Mesh basis; it must be less than `degree`.
`BodyFitMeshSpec.bc_penalty_factor`	Attribute	Positive penalty used to impose essential boundary conditions.
`BodyFitMeshSpec.tessellation_size`	Attribute	Positive target tessellation size, or `None` for the default.
`BodyFitMeshSpec.meshType`	Attribute	Persistent Mesh discriminator for this specification.
`MeshSpec`	Attribute	Any supported portable Mesh specification.
`ElementSizeSizing.to_dict`	Instance method	Serialize this specification to a versioned dictionary.
`ElementSizeSizing.to_json`	Instance method	Serialize this specification to a versioned JSON document.
`ElementSizeSizing.from_dict`	Class method	Create a validated sizing specification from a mapping payload.
`ElementSizeSizing.from_json`	Class method	Create a validated specification from a JSON document.
`ElementSizeSizing.copy_with`	Instance method	Return a validated copy with the supplied fields replaced.
`ElementSizeSizing.validate`	Instance method	Return validation issues for the element-size specification.
`ElementIntervalCountByAxisSizing.to_dict`	Instance method	Serialize this specification to a versioned dictionary.
`ElementIntervalCountByAxisSizing.to_json`	Instance method	Serialize this specification to a versioned JSON document.

Name	Kind	Description
`ElementIntervalCountByAxisSizing.from_dict`	Class method	Create a validated sizing specification from a mapping payload.
`ElementIntervalCountByAxisSizing.from_json`	Class method	Create a validated specification from a JSON document.
`ElementIntervalCountByAxisSizing.with_`	Instance method	Return a validated copy with the supplied fields replaced.
`ElementIntervalCountByAxisSizing.validate`	Instance method	Return validation issues for the per-axis interval counts.
`ElementIntervalCountSizing.to_dict`	Instance method	Serialize this specification to a versioned dictionary.
`ElementIntervalCountSizing.to_json`	Instance method	Serialize this specification to a versioned JSON document.
`ElementIntervalCountSizing.from_dict`	Class method	Create a validated sizing specification from a mapping payload.
`ElementIntervalCountSizing.from_json`	Class method	Create a validated specification from a JSON document.
`ElementIntervalCountSizing.with_`	Instance method	Return a validated copy with the supplied fields replaced.
`ElementIntervalCountSizing.validate`	Instance method	Return validation issues for the interval counts.
`Frame.to_dict`	Instance method	Serialize this specification to a versioned dictionary.
`Frame.to_json`	Instance method	Serialize this specification to a versioned JSON document.

Name	Kind	Description
`Frame.from_dict`	Class method	Create a validated specification from a mapping payload.
`Frame.from_json`	Class method	Create a validated specification from a JSON document.
`Frame.copy_with`	Instance method	Return a validated copy with the supplied fields replaced.
`Frame.validate`	Instance method	Return validation issues for the frame basis.
`ImmersedRectilinearMeshSpec.to_dict`	Instance method	Serialize this specification to a versioned dictionary.
`ImmersedRectilinearMeshSpec.to_json`	Instance method	Serialize this specification to a versioned JSON document.
`ImmersedRectilinearMeshSpec.from_dict`	Class method	Create a validated specification from a mapping payload.
`ImmersedRectilinearMeshSpec.from_json`	Class method	Create a validated specification from a JSON document.
`ImmersedRectilinearMeshSpec.copy_with`	Instance method	Return a validated copy with the supplied fields replaced.
`ImmersedRectilinearMeshSpec.validate`	Instance method	Return validation issues for the immersed rectilinear Mesh.
`ImmersedRectilinearMeshSpec.has_explicit_frame`	Instance method	Return whether both a frame and its extents are defined.
`BodyFitMeshSpec.to_dict`	Instance method	Serialize this specification to a versioned dictionary.

Name	Kind	Description
`BodyFitMeshSpec.to_json`	Instance method	Serialize this specification to a versioned JSON document.
`BodyFitMeshSpec.from_dict`	Class method	Create a validated specification from a mapping payload.
`BodyFitMeshSpec.from_json`	Class method	Create a validated specification from a JSON document.
`BodyFitMeshSpec.copy_with`	Instance method	Return a validated copy with the supplied fields replaced.
`BodyFitMeshSpec.validate`	Instance method	Return validation issues for the body-fit Mesh.
`MeshType`	Class	Persistent Mesh kinds supported by the portable API.
`ElementSizeSizing`	Class	Specify rectilinear Mesh sizing with one element size per axis.
`ElementIntervalCountByAxisSizing`	Class	Specify rectilinear Mesh sizing with an interval count for each axis.
`ElementIntervalCountSizing`	Class	Specify rectilinear Mesh sizing with a three-axis interval-count tuple.
`Frame`	Class	Define the origin and orthonormal basis of a local Mesh frame.
`ImmersedRectilinearMeshSpec`	Class	Define a portable immersed rectilinear Mesh.
`BodyFitMeshSpec`	Class	Define a portable body-fit Mesh.

2.1 `MeshType`

Object path: `coreform.mesh.specs.MeshType`

Persistent Mesh kinds supported by the portable API.

2.1.a Attributes

Name	Type	Value	Description
`IMMERSED_RECTILINEAR`	`MeshType`	`'ImmersedRectilinearMesh'`	Immersed rectilinear Mesh specification.
`BODY_FIT`	`MeshType`	`'BodyFitMesh'`	Body-fit Mesh specification.

2.2 `ElementSizeSizing`

Object path: `coreform.mesh.specs.ElementSizeSizing`

Specify rectilinear Mesh sizing with one element size per axis.

Call syntax

```
result = ElementSizeSizing(element_size=element_size)
```

Signatures

```
ElementSizeSizing(element_size: Float3)
```

Parameters

Name	Type	Required	Default	Description
`element_size`	`Float3`	Yes	—	Positive target element size along each frame axis.

2.2.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`element_size`	`Float3`	—	Positive target element size along each frame axis.

Name	Type	Value	Description
`sizing_type`	`str`	`'ElementSizeSizing'`	Serialized discriminator for this sizing form.

2.2.b `to_dict`

Defined on: `coreform.mesh.specs.ElementSizeSizing.to_dict`

Description: Serialize this specification to a versioned dictionary.

Call syntax

```
result = spec.to_dict()
```

Returns

Type	Description
`Dict[str, Any]`	Canonical specification fields with top-level schema metadata.

Signatures

```
to_dict() -> Dict[str, Any]
```

Parameters

None.

2.2.c `to_json`

Defined on: `coreform.mesh.specs.ElementSizeSizing.to_json`

Description: Serialize this specification to a versioned JSON document.

Call syntax

```
result = spec.to_json()
```

Returns

Type	Description
`str`	Versioned JSON representation of this specification.

Signatures

```
to_json(indent: Optional[int] = 2) -> str
```

Parameters

Name	Type	Required	Default	Description
`indent`	`Optional[int]`	No	`2`	Number of spaces used to indent the JSON, or `None` for compact JSON.

2.2.d `from_dict`

Defined on: `coreform.mesh.specs.ElementSizeSizing.from_dict`

Description: Create a validated sizing specification from a mapping payload.

Call syntax

```
result = ElementSizeSizing.from_dict(payload=payload)
```

Returns

Type	Description
`_SizingSpecT`	Create a validated sizing specification from a mapping payload.

Signatures

```
from_dict(payload: Mapping[str, Any]) -> ElementSizeSizing
```

Parameters

Name	Type	Required	Default	Description
`payload`	`Mapping[str, Any]`	Yes	—	Sizing fields and optional discriminator to validate and load.

2.2.e `from_json`

Defined on: `coreform.mesh.specs.ElementSizeSizing.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

```
result = ElementSizeSizing.from_json(text=text)
```

Returns

Type	Description
`CoreformSpec`	Validated instance of the concrete specification class.

Signatures

```
from_json(text: str) -> ElementSizeSizing
```

Parameters

Name	Type	Required	Default	Description
`text`	`str`	Yes	—	Versioned JSON specification document.

2.2.f `copy_with`

Defined on: `coreform.mesh.specs.ElementSizeSizing.copy_with`

Description: Return a validated copy with the supplied fields replaced.

Call syntax

```
result = spec.copy_with(...)
```

Returns

Type	Description
`CoreformSpec`	Validated copy of the same concrete specification type.

Signatures

```
copy_with(**kwargs: Any) -> CoreformSpec
```

Parameters

Name	Type	Required	Default	Description
`kwargs`	`Any`	No	—	Specification fields and their replacement values.

2.2.g `validate`

Defined on: `coreform.mesh.specs.ElementSizeSizing.validate`

Description: Return validation issues for the element-size specification.

Call syntax

```
result = spec.validate()
```

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	Return validation issues for the element-size specification.

Signatures

```
validate() -> Tuple[ValidationIssue, ...]
```

Parameters

None.

2.3 `ElementIntervalCountByAxisSizing`

Object path: `coreform.mesh.specs.ElementIntervalCountByAxisSizing`

Specify rectilinear Mesh sizing with an interval count for each axis.

Call syntax

```
result = ElementIntervalCountByAxisSizing(element_interval_count_by_axis=element_interval_count_by
```

Signatures

```
ElementIntervalCountByAxisSizing(element_interval_count_by_axis:    Tuple[int,
int, int])
```

Parameters

Name	Type	Required	Default	Description
`element_interval_count`	`Tuple[int, int, int]`	Yes	—	Positive element-interval count along each frame axis.

2.3.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`element_interval_count`	`Tuple[int, int, int]`	—	Positive element-interval count along each frame axis.
`sizing_type`	`str`	`'ElementIntervalCountByAxisSizing'`	Serialized discriminator for this sizing form.

2.3.b `to_dict`

Defined on: `coreform.mesh.specs.ElementIntervalCountByAxisSizing.to_dict`

Description: Serialize this specification to a versioned dictionary.

Call syntax

```
result = spec.to_dict()
```

Returns

Type	Description
`Dict[str, Any]`	Canonical specification fields with top-level schema metadata.

Signatures

```
to_dict() -> Dict[str, Any]
```

Parameters

None.

2.3.c `to_json`

Defined on: `coreform.mesh.specs.ElementIntervalCountByAxisSizing.to_json`

Description: Serialize this specification to a versioned JSON document.

Call syntax

```
result = spec.to_json()
```

Returns

Type	Description
`str`	Versioned JSON representation of this specification.

Signatures

```
to_json(indent: Optional[int] = 2) -> str
```

Parameters

Name	Type	Required	Default	Description
`indent`	`Optional[int]`	No	`2`	Number of spaces used to indent the JSON, or `None` for compact JSON.

2.3.d `from_dict`

Defined on: `coreform.mesh.specs.ElementIntervalCountByAxisSizing.from_dict`

Description: Create a validated sizing specification from a mapping payload.

Call syntax

```
result = ElementIntervalCountByAxisSizing.from_dict(payload=payload)
```

Returns

Type	Description
`_SizingSpecT`	Create a validated sizing specification from a mapping payload.

Signatures

```
from_dict(payload: Mapping[str, Any]) -> ElementIntervalCountByAxisSizing
```

Parameters

Name	Type	Required	Default	Description
`payload`	`Mapping[str, Any]`	Yes	—	Sizing fields and optional discriminator to validate and load.

2.3.e `from_json`

Defined on: `coreform.mesh.specs.ElementIntervalCountByAxisSizing.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

```
result = ElementIntervalCountByAxisSizing.from_json(text=text)
```

Returns

Type	Description
`CoreformSpec`	Validated instance of the concrete specification class.

Signatures

```
from_json(text: str) -> ElementIntervalCountByAxisSizing
```

Parameters

Name	Type	Required	Default	Description
`text`	`str`	Yes	—	Versioned JSON specification document.

2.3.f `copy_with`

Defined on: `coreform.mesh.specs.ElementIntervalCountByAxisSizing.copy_with`

Description: Return a validated copy with the supplied fields replaced.

Call syntax

```
result = spec.copy_with(...)
```

Returns

Type	Description
`CoreformSpec`	Validated copy of the same concrete specification type.

Signatures

```
copy_with(**kwargs: Any) -> CoreformSpec
```

Parameters

Name	Type	Required	Default	Description
`kwargs`	`Any`	No	—	Specification fields and their replacement values.

2.3.g `validate`

Defined on: `coreform.mesh.specs.ElementIntervalCountByAxisSizing.validate`

Description: Return validation issues for the per-axis interval counts.

Call syntax

```
result = spec.validate()
```

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	Return validation issues for the per-axis interval counts.

Signatures

```
validate() -> Tuple[ValidationIssue, ...]
```

Parameters

None.

2.4 `ElementIntervalCountSizing`

Object path: `coreform.mesh.specs.ElementIntervalCountSizing`

Specify rectilinear Mesh sizing with a three-axis interval-count tuple.

Call syntax

```
result
= ElementIntervalCountSizing(element_interval_count=element_interval_count)
```

Signatures

```
ElementIntervalCountSizing(element_interval_count: Tuple[int, int, int])
```

Parameters

Name	Type	Required	Default	Description
`element_interval_count`	`Tuple[int, int, int]`	Yes	—	Positive element-interval count for the three frame axes.

2.4.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`element_interval_count`	`Tuple[int, int, int]`	—	Positive element-interval count for the three frame axes.
`sizing_type`	`str`	`'ElementIntervalCountSizing'`	Serialized discriminator for this sizing form.

2.4.b `to_dict`

Defined on: `coreform.mesh.specs.ElementIntervalCountSizing.to_dict`

Description: Serialize this specification to a versioned dictionary.

Call syntax

```
result = spec.to_dict()
```

Returns

Type	Description
`Dict[str, Any]`	Canonical specification fields with top-level schema metadata.

Signatures

```
to_dict() -> Dict[str, Any]
```

Parameters

None.

2.4.c `to_json`

Defined on: `coreform.mesh.specs.ElementIntervalCountSizing.to_json`

Description: Serialize this specification to a versioned JSON document.

Call syntax

```
result = spec.to_json()
```

Returns

Type	Description
`str`	Versioned JSON representation of this specification.

Signatures

```
to_json(indent: Optional[int] = 2) -> str
```

Parameters

Name	Type	Required	Default	Description
`indent`	`Optional[int]`	No	`2`	Number of spaces used to indent the JSON, or `None` for compact JSON.

2.4.d `from_dict`

Defined on: `coreform.mesh.specs.ElementIntervalCountSizing.from_dict`

Description: Create a validated sizing specification from a mapping payload.

Call syntax

```
result = ElementIntervalCountSizing.from_dict(payload=payload)
```

Returns

Type	Description
`_SizingSpecT`	Create a validated sizing specification from a mapping payload.

Signatures

```
from_dict(payload: Mapping[str, Any]) -> ElementIntervalCountSizing
```

Parameters

Name	Type	Required	Default	Description
`payload`	`Mapping[str, Any]`	Yes	—	Sizing fields and optional discriminator to validate and load.

2.4.e `from_json`

Defined on: `coreform.mesh.specs.ElementIntervalCountSizing.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

```
result = ElementIntervalCountSizing.from_json(text=text)
```

Returns

Type	Description
`CoreformSpec`	Validated instance of the concrete specification class.

Signatures

```
from_json(text: str) -> ElementIntervalCountSizing
```

Parameters

Name	Type	Required	Default	Description
`text`	`str`	Yes	—	Versioned JSON specification document.

2.4.f `copy_with`

Defined on: `coreform.mesh.specs.ElementIntervalCountSizing.copy_with`

Description: Return a validated copy with the supplied fields replaced.

Call syntax

```
result = spec.copy_with(...)
```

Returns

Type	Description
`CoreformSpec`	Validated copy of the same concrete specification type.

Signatures

```
copy_with(**kwargs: Any) -> CoreformSpec
```

Parameters

Name	Type	Required	Default	Description
`kwargs`	`Any`	No	—	Specification fields and their

Name	Type	Required	Default	Description
				replacement values.

2.4.g `validate`

Defined on: `coreform.mesh.specs.ElementIntervalCountSizing.validate`

Description: Return validation issues for the interval counts.

Call syntax

```
result = spec.validate()
```

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	Return validation issues for the interval counts.

Signatures

```
validate() -> Tuple[ValidationIssue, ...]
```

Parameters

None.

2.5 `RectilinearSizing`

Object path: `coreform.mesh.specs.RectilinearSizing`

Type: `Union[ElementSizeSizing, ElementIntervalCountByAxisSizing, ElementIntervalCountSizing]`

Any supported rectilinear Mesh sizing specification.

2.6 `Frame`

Object path: `coreform.mesh.specs.Frame`

Define the origin and orthonormal basis of a local Mesh frame.

Call syntax

```
result = Frame(origin=origin, basis=basis)
```

Signatures

```
Frame(origin: Vector3D, basis: Tuple[Vector3D, Vector3D, Vector3D])
```

Parameters

Name	Type	Required	Default	Description
`origin`	`Vector3D`	Yes	—	Origin of the local frame in global coordinates.
`basis`	`Tuple[Vector3D, Vector3D, Vector3D]`	Yes	—	Right-handed orthonormal basis vectors stored as rows.

2.6.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`origin`	`Vector3D`	—	Origin of the local frame in global coordinates.
`basis`	`Tuple[Vector3D, Vector3D, Vector3D]`	—	Right-handed orthonormal basis vectors stored as rows.

2.6.b `to_dict`

Defined on: `coreform.mesh.specs.Frame.to_dict`

Description: Serialize this specification to a versioned dictionary.

Call syntax

```
result = spec.to_dict()
```

Returns

Type	Description
`Dict[str, Any]`	Canonical specification fields with top-level schema metadata.

Signatures

```
to_dict() -> Dict[str, Any]
```

Parameters

None.

2.6.c `to_json`

Defined on: `coreform.mesh.specs.Frame.to_json`

Description: Serialize this specification to a versioned JSON document.

Call syntax

```
result = spec.to_json()
```

Returns

Type	Description
`str`	Versioned JSON representation of this specification.

Signatures

```
to_json(indent: Optional[int] = 2) -> str
```

Parameters

Name	Type	Required	Default	Description
`indent`	`Optional[int]`	No	`2`	Number of spaces used to indent the JSON, or `None` for compact JSON.

2.6.d `from_dict`

Defined on: `coreform.mesh.specs.Frame.from_dict`

Description: Create a validated specification from a mapping payload.

Call syntax

```
result = Frame.from_dict(payload=payload)
```

Returns

Type	Description
`CoreformSpec`	Validated instance of the concrete specification class.

Signatures

```
from_dict(payload: Mapping[str, Any]) -> Frame
```

Parameters

Name	Type	Required	Default	Description
`payload`	`Mapping[str, Any]`	Yes	—	Versioned specification fields to validate and load.

2.6.e `from_json`

Defined on: `coreform.mesh.specs.Frame.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

```
result = Frame.from_json(text=text)
```

Returns

Type	Description
`CoreformSpec`	Validated instance of the concrete specification class.

Signatures

```
from_json(text: str) -> Frame
```

Parameters

Name	Type	Required	Default	Description
`text`	`str`	Yes	—	Versioned JSON specification document.

2.6.f `copy_with`

Defined on: `coreform.mesh.specs.Frame.copy_with`

Description: Return a validated copy with the supplied fields replaced.

Call syntax

```
result = spec.copy_with(...)
```

Returns

Type	Description
`CoreformSpec`	Validated copy of the same concrete specification type.

Signatures

```
copy_with(**kwargs: Any) -> CoreformSpec
```

Parameters

Name	Type	Required	Default	Description
`kwargs`	`Any`	No	—	Specification fields and their replacement values.

2.6.g `validate`

Defined on: `coreform.mesh.specs.Frame.validate`

Description: Return validation issues for the frame basis.

Call syntax

```
result = spec.validate()
```

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	Return validation issues for the frame basis.

Signatures

```
validate() -> Tuple[ValidationIssue, ...]
```

Parameters

None.

2.7 `ImmersedRectilinearMeshSpec`

Object path: `coreform.mesh.specs.ImmersedRectilinearMeshSpec`

Define a portable immersed rectilinear Mesh.

Call syntax

```
result = ImmersedRectilinearMeshSpec(sizing=sizing)
```

Signatures

```
ImmersedRectilinearMeshSpec(degree: int = 1, continuity: int = 0,
bc_penalty_value: Optional[float] = None, tessellation_size: Optional[float]
= None, sizing: Optional[RectilinearSizing] = None, element_layout:
Optional[Tuple[str, str, str]] = None, small_cell_volume_ratio: float = 0.2,
padding: Optional[Tuple[int, int, int]] = None, frame: Optional[Frame] = None,
extents: Optional[Tuple[Float3, Float3]] = None)
```

Parameters

Name	Type	Required	Default	Description
`degree`	`int`	No	`1`	Polynomial degree of the Mesh basis.
`continuity`	`int`	No	`0`	Continuity of the Mesh basis; it must be less than `degree`.

Name	Type	Required	Default	Description
`bc_penalty_value`	`Optional[float]`	No	`None`	Positive penalty used to impose essential boundary conditions.
`tessellation_size`	`Optional[float]`	No	`None`	Positive target tessellation size, or `None` for the default.
`sizing`	`Optional[RectilinearSizing]`	Yes	—	Required element-size or interval-count sizing specification.
`element_layout`	`Optional[Tuple[str, str, str]]`	No	`None`	Per-axis `edge_centered` or `node_centered` layout choices.
`small_cell_volume_ratio`	`float`	No	`0.2`	Small-cell threshold in the half-open interval `[0, 0.5)`.
`padding`	`Optional[Tuple[int, int, int]]`	No	`None`	Nonnegative cell padding along each frame axis.
`frame`	`Optional[Frame]`	No	`None`	Explicit local frame; provide it together with `extents`.
`extents`	`Optional[Tuple[Float3, Float3]]`	No	`None`	Increasing minimum and maximum cor-

Name	Type	Required	Default	Description
				ners in frame coordinates.

2.7.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`degree`	`int`	`1`	Polynomial degree of the Mesh basis.
`continuity`	`int`	`0`	Continuity of the Mesh basis; it must be less than `degree`.
`bc_penalty_value`	`Optional[float]`	`None`	Positive penalty used to impose essential boundary conditions.
`tessellation_size`	`Optional[float]`	`None`	Positive target tessellation size, or `None` for the default.
`sizing`	`Optional[RectilinearSizing]`	`field(default=None, metadata={'required': True})`	Required element-size or interval-count sizing specification.
`element_layout`	`Optional[Tuple[str, str, str]]`	`None`	Per-axis `edge_centered` or `node_centered` layout choices.
`small_cell_volume_ratio`	`float`	`0.2`	Small-cell threshold in the half-open interval `[0, 0.5)`.
`padding`	`Optional[Tuple[int, int, int]]`	`None`	Nonnegative cell padding along each frame axis.

Name	Type	Value	Description
`frame`	`Optional[Frame]`	`None`	Explicit local frame; provide it together with `extents`.
`extents`	`Optional[Tuple[Float3, Float3]]`	`field(default=None, metadata={'camel_case': 'frameExtents'})`	Increasing minimum and maximum corners in frame coordinates.
`meshType`	`MeshType`	`MeshType.IMMERSED_RECTILINEAR`	Persistent Mesh discriminator for this specification.

2.7.b `to_dict`

Defined on: `coreform.mesh.specs.ImmersedRectilinearMeshSpec.to_dict`

Description: Serialize this specification to a versioned dictionary.

Call syntax

```
result = spec.to_dict()
```

Returns

Type	Description
`Dict[str, Any]`	Canonical specification fields with top-level schema metadata.

Signatures

```
to_dict() -> Dict[str, Any]
```

Parameters

None.

2.7.c `to_json`

Defined on: `coreform.mesh.specs.ImmersedRectilinearMeshSpec.to_json`

Description: Serialize this specification to a versioned JSON document.

Call syntax

```
result = spec.to_json()
```

Returns

Type	Description
`str`	Versioned JSON representation of this specification.

Signatures

```
to_json(indent: Optional[int] = 2) -> str
```

Parameters

Name	Type	Required	Default	Description
`indent`	`Optional[int]`	No	`2`	Number of spaces used to indent the JSON, or `None` for compact JSON.

2.7.d `from_dict`

Defined on: `coreform.mesh.specs.ImmersedRectilinearMeshSpec.from_dict`

Description: Create a validated specification from a mapping payload.

Call syntax

```
result = ImmersedRectilinearMeshSpec.from_dict(payload=payload)
```

Returns

Type	Description
`CoreformSpec`	Validated instance of the concrete specification class.

Signatures

```
from_dict(payload: Mapping[str, Any]) -> ImmersedRectilinearMeshSpec
```

Parameters

Name	Type	Required	Default	Description
`payload`	`Mapping[str, Any]`	Yes	—	Versioned specification fields to validate and load.

2.7.e `from_json`

Defined on: `coreform.mesh.specs.ImmersedRectilinearMeshSpec.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

```
result = ImmersedRectilinearMeshSpec.from_json(text=text)
```

Returns

Type	Description
`CoreformSpec`	Validated instance of the concrete specification class.

Signatures

```
from_json(text: str) -> ImmersedRectilinearMeshSpec
```

Parameters

Name	Type	Required	Default	Description
`text`	`str`	Yes	—	Versioned JSON specification document.

2.7.f `copy_with`

Defined on: `coreform.mesh.specs.ImmersedRectilinearMeshSpec.copy_with`

Description: Return a validated copy with the supplied fields replaced.

Call syntax

```
result = spec.copy_with(...)
```

Returns

Type	Description
`CoreformSpec`	Validated copy of the same concrete specification type.

Signatures

```
copy_with(**kwargs: Any) -> CoreformSpec
```

Parameters

Name	Type	Required	Default	Description
`kwargs`	`Any`	No	—	Specification fields and their replacement values.

2.7.g `validate`

Defined on: `coreform.mesh.specs.ImmersedRectilinearMeshSpec.validate`

Description: Return validation issues for the immersed rectilinear Mesh.

Call syntax

```
result = spec.validate()
```

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	Return validation issues for the immersed rectilinear Mesh.

Signatures

```
validate() -> Tuple[ValidationIssue, ...]
```

Parameters

None.

2.7.h `has_explicit_frame`

Defined on: `coreform.mesh.specs.ImmersedRectilinearMeshSpec.has_explicit_frame`

Description: Return whether both a frame and its extents are defined.

Call syntax

```
result = spec.has_explicit_frame()
```

Returns

Type	Description
`bool`	Return whether both a frame and its extents are defined.

Signatures

```
has_explicit_frame() -> bool
```

Parameters

None.

2.8 `BodyFitMeshSpec`

Object path: `coreform.mesh.specs.BodyFitMeshSpec`

Define a portable body-fit Mesh.

Call syntax

```
result = BodyFitMeshSpec()
```

Signatures

```
BodyFitMeshSpec(degree: int = 1, continuity: int = 0, bc_penalty_value:
Optional[float] = None, tessellation_size: Optional[float] = None)
```

Parameters

Name	Type	Required	Default	Description
`degree`	`int`	No	`1`	Polynomial degree of the Mesh basis.
`continuity`	`int`	No	`0`	Continuity of the Mesh basis; it must be less than `degree`.

Name	Type	Required	Default	Description
`bc_penalty_value`	`Optional[float]`	No	`None`	Positive penalty used to impose essential boundary conditions.
`tessellation_size`	`Optional[float]`	No	`None`	Positive target tessellation size, or `None` for the default.

2.8.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`degree`	`int`	`1`	Polynomial degree of the Mesh basis.
`continuity`	`int`	`0`	Continuity of the Mesh basis; it must be less than `degree`.
`bc_penalty_value`	`Optional[float]`	`None`	Positive penalty used to impose essential boundary conditions.
`tessellation_size`	`Optional[float]`	`None`	Positive target tessellation size, or `None` for the default.
`meshType`	`MeshType`	`MeshType.BODY_FIT`	Persistent Mesh discriminator for this specification.

2.8.b `to_dict`

Defined on: `coreform.mesh.specs.BodyFitMeshSpec.to_dict`

Description: Serialize this specification to a versioned dictionary.

Call syntax

```
result = spec.to_dict()
```

Returns

Type	Description
`Dict[str, Any]`	Canonical specification fields with top-level schema metadata.

Signatures

```
to_dict() -> Dict[str, Any]
```

Parameters

None.

2.8.c `to_json`

Defined on: `coreform.mesh.specs.BodyFitMeshSpec.to_json`

Description: Serialize this specification to a versioned JSON document.

Call syntax

```
result = spec.to_json()
```

Returns

Type	Description
`str`	Versioned JSON representation of this specification.

Signatures

```
to_json(indent: Optional[int] = 2) -> str
```

Parameters

Name	Type	Required	Default	Description
`indent`	`Optional[int]`	No	`2`	Number of spaces used to indent the JSON, or `None`

Name	Type	Required	Default	Description
				for compact JSON.

2.8.d `from_dict`

Defined on: `coreform.mesh.specs.BodyFitMeshSpec.from_dict`

Description: Create a validated specification from a mapping payload.

Call syntax

```
result = BodyFitMeshSpec.from_dict(payload=payload)
```

Returns

Type	Description
`CoreformSpec`	Validated instance of the concrete specification class.

Signatures

```
from_dict(payload: Mapping[str, Any]) -> BodyFitMeshSpec
```

Parameters

Name	Type	Required	Default	Description
`payload`	`Mapping[str, Any]`	Yes	—	Versioned specification fields to validate and load.

2.8.e `from_json`

Defined on: `coreform.mesh.specs.BodyFitMeshSpec.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

```
result = BodyFitMeshSpec.from_json(text=text)
```

Returns

Type	Description
`CoreformSpec`	Validated instance of the concrete specification class.

Signatures

```
from_json(text: str) -> BodyFitMeshSpec
```

Parameters

Name	Type	Required	Default	Description
`text`	`str`	Yes	—	Versioned JSON specification document.

2.8.f `copy_with`

Defined on: `coreform.mesh.specs.BodyFitMeshSpec.copy_with`

Description: Return a validated copy with the supplied fields replaced.

Call syntax

```
result = spec.copy_with(...)
```

Returns

Type	Description
`CoreformSpec`	Validated copy of the same concrete specification type.

Signatures

```
copy_with(**kwargs: Any) -> CoreformSpec
```

Parameters

Name	Type	Required	Default	Description
`kwargs`	`Any`	No	—	Specification fields and their replacement values.

2.8.g `validate`

Defined on: `coreform.mesh.specs.BodyFitMeshSpec.validate`

Description: Return validation issues for the body-fit Mesh.

Call syntax

```
result = spec.validate()
```

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	Return validation issues for the body-fit Mesh.

Signatures

```
validate() -> Tuple[ValidationIssue, ...]
```

Parameters

None.

2.9 `MeshSpec`

Object path: `coreform.mesh.specs.MeshSpec`

Type: `Union[ImmersedRectilinearMeshSpec, BodyFitMeshSpec]`

Any supported portable Mesh specification.