coreform.probe

Table of contents

1 model.py 2
1.1 IGAProbe 3
1.1.a Attributes 4
1.1.b point 4
1.1.c line 6
1.1.d multi_point 7
1.1.e from_spec 8
1.1.f to_spec 9
1.1.g copy_with 9
1.1.h is_point 12
1.1.i is_line 13
1.1.j is_multi_point 13
2 specs.py 14
2.1 ProbeType 18
2.1.a Attributes 18
2.2 ProbeRegionKind 19
2.2.a Attributes 19
2.3 ProbeRegion 19
2.3.a Attributes 20
2.3.b validate 20
2.3.c to_dict 20
2.3.d to_json 21
2.3.e from_json 22
2.3.f copy_with 22
2.3.g instances 23
2.3.h sets 23
2.3.i parts 24
2.3.j from_dict 24
2.4 PointProbeSpec 25
2.4.a Attributes 26
2.4.b validate 26
2.4.c to_dict 27
2.4.d to_json 27
2.4.e from_dict 28
2.4.f from_json 29

2.4.g copy_with 29
2.5 LineProbeSpec 30
2.5.a Attributes 31
2.5.b to_dict 32
2.5.c to_json 32
2.5.d from_dict 33
2.5.e from_json 33
2.5.f copy_with 34
2.5.g validate 35
2.6 MultiPointProbeSpec 35
2.6.a Attributes 36
2.6.b validate 36
2.6.c to_dict 37
2.6.d to_json 37
2.6.e from_dict 38
2.6.f from_json 39
2.6.g copy_with 39
2.7 ProbeSpec 40

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

Directory tree

- `probe/`

1 model.py

Module path: `coreform.probe.model`

Available API members

Name	Kind	Description
`IGAProbe.spec`	Attribute	Wrapped portable Probe specification.
`IGAProbe.name`	Attribute	Return this Probe's name.
`IGAProbe.region`	Attribute	Return this Probe's target region.
`IGAProbe.variables`	Attribute	Return this Probe's output-variable names.
`IGAProbe.probe_type`	Attribute	Return this Probe's type.

Name	Kind	Description
`IGAProbe.point`	Class method	Create a portable point Probe.
`IGAProbe.line`	Class method	Create a portable line Probe.
`IGAProbe.multi_point`	Class method	Create a portable multi-point Probe.
`IGAProbe.from_spec`	Class method	Wrap an existing portable Probe specification.
`IGAProbe.to_spec`	Instance method	Return the wrapped portable Probe specification.
`IGAProbe.copy_with`	Instance method	Return a Probe with the supplied specification fields replaced.
`IGAProbe.is_point`	Instance method	Return whether this Probe wraps a point-Probe specification.
`IGAProbe.is_line`	Instance method	Return whether this Probe wraps a line-Probe specification.
`IGAProbe.is_multi_point`	Instance method	Return whether this Probe wraps a multi-point-Probe specification.
`IGAProbe`	Class	Portable Probe model wrapping one concrete Probe specification.

1.1 `IGAProbe`

Object path: `coreform.probe.model.IGAProbe`

Portable Probe model wrapping one concrete Probe specification.

Call syntax

```
result = IGAProbe(spec=spec)
```

Signatures

```
IGAProbe(spec: ProbeSpec)
```

Parameters

Name	Type	Required	Default	Description
`spec`	`ProbeSpec`	Yes	—	Concrete portable Probe specification to wrap.

Example

```
>>> probe = IGAProbe.point(
...     name="Probe-1",
...     region=ProbeRegion.instances(("PART-1-1",)),
...     location=(0.0, 0.0, 0.0),
... )
```

1.1.a Attributes

Name	Type	Value	Description
`spec`	`ProbeSpec`	—	Wrapped portable Probe specification.
`name`	`str`	—	Return this Probe's name.
`region`	`ProbeRegion`	—	Return this Probe's target region.
`variables`	`Tuple[str, ...]`	—	Return this Probe's output-variable names.
`probe_type`	`ProbeType`	—	Return this Probe's type.

1.1.b `point`

Defined on: `coreform.probe.model.IGAProbe.point`

Description: Create a portable point Probe.

Call syntax

```
result = IGAProbe.point(spec)
result = IGAProbe.point(name=name, region=region, location=location)
```

Returns

Type	Description
`IGAProbe`	Create a portable point Probe.

Signatures

```
point(spec: PointProbeSpec, /) -> IGAProbe
point(name: str = None, region: ProbeRegion = None, variables: Tuple[str, ...]
= tuple(), location: Point3D = None) -> IGAProbe
```

Parameters

For `point(spec: PointProbeSpec, /) -> IGAProbe`:

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

For `point(name: str = None, region: ProbeRegion = None, variables: Tuple[str, ...] = tuple(), location: Point3D = None) -> IGAProbe`:

Name	Type	Required	Default	Description
`name`	`str`	Yes	—	Unique, non-empty Probe name.
`region`	`ProbeRegion`	Yes	—	Portable target selection.
`variables`	`Tuple[str, ...]`	No	`()`	Output-variable names; the default is an empty tuple.
`location`	`Point3D`	Yes	—	Evaluation location.

1.1.c `line`

Defined on: `coreform.probe.model.IGAProbe.line`

Description: Create a portable line Probe.

Call syntax

```
result = IGAProbe.line(spec)
result = IGAProbe.line(name=name, region=region, start_location=start_location, stop_location=stop_location)
```

Returns

Type	Description
`IGAProbe`	Create a portable line Probe.

Signatures

```
line(spec: LineProbeSpec, /) -> IGAProbe
line(name: str = None, region: ProbeRegion = None, variables: Tuple[str, ...] = tuple(), start_location: Point3D = None, stop_location: Point3D = None, num_eval_points: int = 2) -> IGAProbe
```

Parameters

For `line(spec: LineProbeSpec, /) -> IGAProbe`:

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

For `line(name: str = None, region: ProbeRegion = None, variables: Tuple[str, ...] = tuple(), start_location: Point3D = None, stop_location: Point3D = None, num_eval_points: int = 2) -> IGAProbe`:

Name	Type	Required	Default	Description
`name`	`str`	Yes	—	Unique, non-empty Probe name.
`region`	`ProbeRegion`	Yes	—	Portable target selection.

Name	Type	Required	Default	Description
`variables`	`Tuple[str, ...]`	No	`()`	Output-variable names; the default is an empty tuple.
`start_location`	`Point3D`	Yes	—	First endpoint of the line segment.
`stop_location`	`Point3D`	Yes	—	Second endpoint; it must differ from start_location.
`num_eval_points`	`int`	No	`2`	Number of evaluation points, including endpoints; it must be at least 2.

1.1.d `multi_point`

Defined on: `coreform.probe.model.IGAProbe.multi_point`

Description: Create a portable multi-point Probe.

Call syntax

```
result = IGAProbe.multi_point(spec)
result = IGAProbe.multi_point(name=name, region=region, locations=locations)
```

Returns

Type	Description
`IGAProbe`	Create a portable multi-point Probe.

Signatures

```
multi_point(spec: MultiPointProbeSpec, /) -> IGAProbe
multi_point(name: str = None, region: ProbeRegion = None, variables:
Tuple[str, ...] = tuple(), locations: Tuple[Point3D, ...] = tuple()) -> IGAProbe
```

Parameters

For `multi_point(spec: MultiPointProbeSpec, /) -> IGAProbe`:

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

For `multi_point(name: str = None, region: ProbeRegion = None, variables: Tuple[str, ...] = tuple(), locations: Tuple[Point3D, ...] = tuple()) -> IGAProbe`:

Name	Type	Required	Default	Description
`name`	`str`	Yes	—	Unique, non-empty Probe name.
`region`	`ProbeRegion`	Yes	—	Portable target selection.
`variables`	`Tuple[str, ...]`	No	`()`	Output-variable names; the default is an empty tuple.
`locations`	`Tuple[Point3D, ...]`	Yes	—	One or more explicit evaluation locations.

1.1.e `from_spec`

Defined on: `coreform.probe.model.IGAProbe.from_spec`

Description: Wrap an existing portable Probe specification.

Call syntax

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

Returns

Type	Description
`IGAProbe`	Wrap an existing portable Probe specification.

Signatures

```
from_spec(spec: ProbeSpec) -> 'IGAProbe'
```

Parameters

Name	Type	Required	Default	Description
`spec`	`ProbeSpec`	Yes	—	Concrete portable Probe specification to wrap.

1.1.f `to_spec`

Defined on: `coreform.probe.model.IGAProbe.to_spec`

Description: Return the wrapped portable Probe specification.

Call syntax

```
result = probe.to_spec()
```

Returns

Type	Description
`ProbeSpec`	Return the wrapped portable Probe specification.

Signatures

```
to_spec() -> ProbeSpec
```

Parameters

None.

1.1.g `copy_with`

Defined on: `coreform.probe.model.IGAProbe.copy_with`

Description: Return a Probe with the supplied specification fields replaced.

Call syntax

```
result = probe.copy_with(...)
```

Returns

Type	Description
`IGAProbe`	Return a Probe with the supplied specification fields replaced.

Signatures

```
copy_with(*, name: str = ..., region: ProbeRegion = ..., variables:
Tuple[str, ...] = ..., location: Point3D = ...) -> IGAProbe
copy_with(*, name: str = ..., region: ProbeRegion = ..., variables:
Tuple[str, ...] = ..., start_location: Point3D = ..., stop_location: Point3D
= ..., num_eval_points: int = ...) -> IGAProbe
copy_with(*, name: str = ..., region: ProbeRegion = ..., variables:
Tuple[str, ...] = ..., locations: Tuple[Point3D, ...] = ...) -> IGAProbe
```

Parameters

For `copy_with(*, name: str = ..., region: ProbeRegion = ..., variables: Tuple[str, ...] = ..., location: Point3D = ...) -> IGAProbe`:

Name	Applies to	Type	Required	Default	Description
`name`	All	`str`	No	Current value	Unique, nonempty Probe name.
`region`	All	`ProbeRegion`	No	Current value	Portable target selection.
`variables`	All	`Tuple[str, ...]`	No	Current value	Output-variable names; the default is an empty tuple.
`location`	`PointProbeSpec`	`Point3D`	No	Current value	Evaluation location.

For `copy_with(*, name: str = ..., region: ProbeRegion = ..., variables: Tuple[str, ...] = ..., start_location: Point3D = ..., stop_location: Point3D = ..., num_eval_points: int = ...) -> IGAProbe`:

Name	Applies to	Type	Required	Default	Description
`name`	All	`str`	No	Current value	Unique, nonempty Probe name.
`region`	All	`ProbeRegion`	No	Current value	Portable target selection.
`variables`	All	`Tuple[str, ...]`	No	Current value	Output-variable names; the default is an empty tuple.
`start_location`	`LineProbeSpec`	`Point3D`	No	Current value	First endpoint of the line segment.
`stop_location`	`LineProbeSpec`	`Point3D`	No	Current value	Second endpoint; it must differ from start_location.
`num_eval_points`	`LineProbeSpec`	`int`	No	Current value	Number of evaluation points, including endpoints; it must be at least 2.

For `copy_with(*, name: str = ..., region: ProbeRegion = ..., variables: Tuple[str, ...] = ..., locations: Tuple[Point3D, ...] = ...) -> IGAProbe`:

Name	Applies to	Type	Required	Default	Description
`name`	All	`str`	No	Current value	Unique, nonempty Probe name.

Name	Applies to	Type	Required	Default	Description
`region`	All	`ProbeRegion`	No	Current value	Portable target selection.
`variables`	All	`Tuple[str, ...]`	No	Current value	Output-variable names; the default is an empty tuple.
`locations`	`MultiPointProbeSpec`	`Tuple[Point3D, ...]`	No	Current value	One or more explicit evaluation locations.

Parameter rules

Rule	Requirement
Applicability	`location` applies only to `PointProbeSpec`.
Applicability	`start_location` applies only to `LineProbeSpec`.
Applicability	`stop_location` applies only to `LineProbeSpec`.
Applicability	`num_eval_points` applies only to `LineProbeSpec`.
Applicability	`locations` applies only to `MultiPointProbeSpec`.

1.1.h `is_point`

Defined on: `coreform.probe.model.IGAProbe.is_point`

Description: Return whether this Probe wraps a point-Probe specification.

Call syntax

```
result = probe.is_point()
```

Returns

Type	Description
`bool`	True if the wrapped specification is a PointProbeSpec; otherwise False.

Signatures

```
is_point() -> bool
```

Parameters

None.

1.1.i `is_line`

Defined on: `coreform.probe.model.IGAProbe.is_line`

Description: Return whether this Probe wraps a line-Probe specification.

Call syntax

```
result = probe.is_line()
```

Returns

Type	Description
`bool`	True if the wrapped specification is a LineProbeSpec; otherwise False.

Signatures

```
is_line() -> bool
```

Parameters

None.

1.1.j `is_multi_point`

Defined on: `coreform.probe.model.IGAProbe.is_multi_point`

Description: Return whether this Probe wraps a multi-point-Probe specification.

Call syntax

```
result = probe.is_multi_point()
```

Returns

Type	Description
`bool`	True if the wrapped specification is a MultiPointProbeSpec; otherwise False.

Signatures

```
is_multi_point() -> bool
```

Parameters

None.

2 specs.py

Module path: `coreform.probe.specs`

Available API members

Name	Kind	Description
`ProbeType.POINT`	Attribute	Point Probe evaluated at one location.
`ProbeType.LINE`	Attribute	Line Probe sampled between two endpoints.
`ProbeType.MULTI_POINT`	Attribute	Multi-point Probe evaluated at explicit locations.
`ProbeRegionKind.INSTANCES`	Attribute	Target named assembly instances.
`ProbeRegionKind.SETS`	Attribute	Target named assembly sets.
`ProbeRegionKind.PARTS`	Attribute	Target named model parts.
`ProbeRegion.schemaVersion`	Attribute	Version of the serialized specification schema.
`ProbeRegion.kind`	Attribute	Target repository kind.
`ProbeRegion.names`	Attribute	One or more target names.
`PointProbeSpec.schemaVersion`	Attribute	Version of the serialized specification schema.
`PointProbeSpec.name`	Attribute	Unique, nonempty Probe name.
`PointProbeSpec.region`	Attribute	Portable target selection.
`PointProbeSpec.variables`	Attribute	Output-variable names; the default is an empty tuple.

Name	Kind	Description
`PointProbeSpec.location`	Attribute	Evaluation location.
`PointProbeSpec.probeType`	Attribute	Point-Probe type identifier.
`LineProbeSpec.schemaVersion`	Attribute	Version of the serialized specification schema.
`LineProbeSpec.name`	Attribute	Unique, nonempty Probe name.
`LineProbeSpec.region`	Attribute	Portable target selection.
`LineProbeSpec.variables`	Attribute	Output-variable names; the default is an empty tuple.
`LineProbeSpec.start_location`	Attribute	First endpoint of the line segment.
`LineProbeSpec.stop_location`	Attribute	Second endpoint; it must differ from start_location.
`LineProbeSpec.num_eval_points`	Attribute	Number of evaluation points, including endpoints; it must be at least 2.
`LineProbeSpec.probeType`	Attribute	Line-Probe type identifier.
`MultiPointProbeSpec.schemaVersion`	Attribute	Version of the serialized specification schema.
`MultiPointProbeSpec.name`	Attribute	Unique, nonempty Probe name.
`MultiPointProbeSpec.region`	Attribute	Portable target selection.
`MultiPointProbeSpec.variables`	Attribute	Output-variable names; the default is an empty tuple.
`MultiPointProbeSpec.locations`	Attribute	One or more explicit evaluation locations.
`MultiPointProbeSpec.probeType`	Attribute	Multi-point-Probe type identifier.
`ProbeSpec`	Attribute	Any supported portable Probe specification.

Name	Kind	Description
`ProbeRegion.validate`	Instance method	Return validation issues for this specification.
`ProbeRegion.to_dict`	Instance method	Serialize this specification to a versioned dictionary.
`ProbeRegion.to_json`	Instance method	Serialize this specification to a versioned JSON document.
`ProbeRegion.from_json`	Class method	Create a validated specification from a JSON document.
`ProbeRegion.copy_with`	Instance method	Return a validated copy with the supplied fields replaced.
`ProbeRegion.instances`	Class method	Create a region targeting named instances.
`ProbeRegion.sets`	Class method	Create a region targeting named sets.
`ProbeRegion.parts`	Class method	Create a region targeting named parts.
`ProbeRegion.from_dict`	Class method	Create a validated Probe region from a mapping payload.
`PointProbeSpec.validate`	Instance method	Return validation issues for this specification.
`PointProbeSpec.to_dict`	Instance method	Serialize this specification to a versioned dictionary.
`PointProbeSpec.to_json`	Instance method	Serialize this specification to a versioned JSON document.
`PointProbeSpec.from_dict`	Class method	Create a validated specification from a mapping payload.

Name	Kind	Description
`PointProbeSpec.from_json`	Class method	Create a validated specification from a JSON document.
`PointProbeSpec.copy_with`	Instance method	Return a validated copy with the supplied fields replaced.
`LineProbeSpec.to_dict`	Instance method	Serialize this specification to a versioned dictionary.
`LineProbeSpec.to_json`	Instance method	Serialize this specification to a versioned JSON document.
`LineProbeSpec.from_dict`	Class method	Create a validated specification from a mapping payload.
`LineProbeSpec.from_json`	Class method	Create a validated specification from a JSON document.
`LineProbeSpec.copy_with`	Instance method	Return a validated copy with the supplied fields replaced.
`LineProbeSpec.validate`	Instance method	Return validation issues for this line-Probe specification.
`MultiPointProbeSpec.validate`	Instance method	Return validation issues for this specification.
`MultiPointProbeSpec.to_dict`	Instance method	Serialize this specification to a versioned dictionary.
`MultiPointProbeSpec.to_json`	Instance method	Serialize this specification to a versioned JSON document.
`MultiPointProbeSpec.from_di`	Class method	Create a validated specification from a mapping payload.

Name	Kind	Description
`MultiPointProbeSpec.from_json`	Class method	Create a validated specification from a JSON document.
`MultiPointProbeSpec.copy_with`	Instance method	Return a validated copy with the supplied fields replaced.
`ProbeType`	Class	Identify the supported Probe geometry types.
`ProbeRegionKind`	Class	Identify the supported Abaqus target-repository kinds.
`ProbeRegion`	Class	Portable selection of one Probe target repository and its names.
`PointProbeSpec`	Class	Portable point-Probe specification.
`LineProbeSpec`	Class	Portable line-Probe specification.
`MultiPointProbeSpec`	Class	Portable multi-point-Probe specification.

2.1 `ProbeType`

Object path: `coreform.probe.specs.ProbeType`

Identify the supported Probe geometry types.

2.1.a Attributes

Name	Type	Value	Description
`POINT`	`ProbeType`	`'PointProbe'`	Point Probe evaluated at one location.
`LINE`	`ProbeType`	`'LineProbe'`	Line Probe sampled between two endpoints.

Name	Type	Value	Description
`MULTI_POINT`	`ProbeType`	`'MultiPointProbe'`	Multi-point Probe evaluated at explicit locations.

2.2 `ProbeRegionKind`

Object path: `coreform.probe.specs.ProbeRegionKind`

Identify the supported Abaqus target-repository kinds.

2.2.a Attributes

Name	Type	Value	Description
`INSTANCES`	`ProbeRegionKind`	`'instances'`	Target named assembly instances.
`SETS`	`ProbeRegionKind`	`'sets'`	Target named assembly sets.
`PARTS`	`ProbeRegionKind`	`'parts'`	Target named model parts.

2.3 `ProbeRegion`

Object path: `coreform.probe.specs.ProbeRegion`

Portable selection of one Probe target repository and its names.

Call syntax

```
result = ProbeRegion(kind=kind, names=names)
```

Signatures

```
ProbeRegion(kind: ProbeRegionKind = None, names: Tuple[str, ...] = tuple())
```

Parameters

Name	Type	Required	Default	Description
`kind`	`ProbeRegionKind`	Yes	—	Target repository kind.
`names`	`Tuple[str, ...]`	Yes	—	One or more target names.

2.3.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`kind`	`ProbeRegionKind`	`None`	Target repository kind.
`names`	`Tuple[str, ...]`	`field(default_factory=tuple, metadata={'required': True})`	One or more target names.

2.3.b `validate`

Defined on: `coreform.probe.specs.ProbeRegion.validate`

Description: Return validation issues for this specification.

Call syntax

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

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	All detected issues, or an empty tuple when the specification is valid.

Signatures

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

Parameters

None.

2.3.c `to_dict`

Defined on: `coreform.probe.specs.ProbeRegion.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.d `to_json`

Defined on: `coreform.probe.specs.ProbeRegion.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.e `from_json`

Defined on: `coreform.probe.specs.ProbeRegion.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

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

Returns

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

Signatures

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

Parameters

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

2.3.f `copy_with`

Defined on: `coreform.probe.specs.ProbeRegion.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 `instances`

Defined on: `coreform.probe.specs.ProbeRegion.instances`

Description: Create a region targeting named instances.

Call syntax

```
result = ProbeRegion.instances(names=names)
```

Returns

Type	Description
`ProbeRegion`	Create a region targeting named instances.

Signatures

```
instances(names: Iterable[str]) -> 'ProbeRegion'
```

Parameters

Name	Type	Required	Default	Description
`names`	`Iterable[str]`	Yes	—	One or more assembly-instance names.

2.3.h `sets`

Defined on: `coreform.probe.specs.ProbeRegion.sets`

Description: Create a region targeting named sets.

Call syntax

```
result = ProbeRegion.sets(names=names)
```

Returns

Type	Description
`ProbeRegion`	Create a region targeting named sets.

Signatures

```
sets(names: Iterable[str]) -> 'ProbeRegion'
```

Parameters

Name	Type	Required	Default	Description
`names`	`Iterable[str]`	Yes	—	One or more assembly-set names.

2.3.i `parts`

Defined on: `coreform.probe.specs.ProbeRegion.parts`

Description: Create a region targeting named parts.

Call syntax

```
result = ProbeRegion.parts(names=names)
```

Returns

Type	Description
`ProbeRegion`	Create a region targeting named parts.

Signatures

```
parts(names: Iterable[str]) -> 'ProbeRegion'
```

Parameters

Name	Type	Required	Default	Description
`names`	`Iterable[str]`	Yes	—	One or more model-part names.

2.3.j `from_dict`

Defined on: `coreform.probe.specs.ProbeRegion.from_dict`

Description: Create a validated Probe region from a mapping payload.

Call syntax

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

Returns

Type	Description
`ProbeRegion`	Create a validated Probe region from a mapping payload.

Signatures

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

Parameters

Name	Type	Required	Default	Description
`payload`	`Mapping[str, Any]`	Yes	—	Region kind and target names to validate and load.

2.4 `PointProbeSpec`

Object path: `coreform.probe.specs.PointProbeSpec`

Portable point-Probe specification.

Call syntax

```
result = PointProbeSpec(name=name, region=region, location=location)
```

Signatures

```
PointProbeSpec(name: str = None, region: ProbeRegion = None, variables:
Tuple[str, ...] = tuple(), location: Point3D = None)
```

Parameters

Name	Type	Required	Default	Description
`name`	`str`	Yes	—	Unique, non-empty Probe name.

Name	Type	Required	Default	Description
`region`	`ProbeRegion`	Yes	—	Portable target selection.
`variables`	`Tuple[str, ...]`	No	`()`	Output-variable names; the default is an empty tuple.
`location`	`Point3D`	Yes	—	Evaluation location.

2.4.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`name`	`str`	`None`	Unique, nonempty Probe name.
`region`	`ProbeRegion`	`None`	Portable target selection.
`variables`	`Tuple[str, ...]`	`()`	Output-variable names; the default is an empty tuple.
`location`	`Point3D`	`None`	Evaluation location.
`probeType`	`ProbeType`	`ProbeType.POINT`	Point-Probe type identifier.

2.4.b `validate`

Defined on: `coreform.probe.specs.PointProbeSpec.validate`

Description: Return validation issues for this specification.

Call syntax

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

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	All detected issues, or an empty tuple when the specification is valid.

Signatures

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

Parameters

None.

2.4.c `to_dict`

Defined on: `coreform.probe.specs.PointProbeSpec.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.d `to_json`

Defined on: `coreform.probe.specs.PointProbeSpec.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.e `from_dict`

Defined on: `coreform.probe.specs.PointProbeSpec.from_dict`

Description: Create a validated specification from a mapping payload.

Call syntax

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

Returns

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

Signatures

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

Parameters

Name	Type	Required	Default	Description
`payload`	`Mapping[str, Any]`	Yes	—	Versioned specification

Name	Type	Required	Default	Description
				fields to validate and load.

2.4.f `from_json`

Defined on: `coreform.probe.specs.PointProbeSpec.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

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

Returns

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

Signatures

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

Parameters

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

2.4.g `copy_with`

Defined on: `coreform.probe.specs.PointProbeSpec.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.

2.5 `LineProbeSpec`

Object path: `coreform.probe.specs.LineProbeSpec`

Portable line-Probe specification.

Call syntax

```
result = LineProbeSpec(name=name, region=region, start_location=start_location, stop_location=stop_location)
```

Signatures

```
LineProbeSpec(name: str = None, region: ProbeRegion = None, variables: Tuple[str, ...] = tuple(), start_location: Point3D = None, stop_location: Point3D = None, num_eval_points: int = 2)
```

Parameters

Name	Type	Required	Default	Description
`name`	`str`	Yes	—	Unique, non-empty Probe name.
`region`	`ProbeRegion`	Yes	—	Portable target selection.
`variables`	`Tuple[str, ...]`	No	`()`	Output-variable names; the default is an empty tuple.

Name	Type	Required	Default	Description
`start_location`	`Point3D`	Yes	—	First endpoint of the line segment.
`stop_location`	`Point3D`	Yes	—	Second endpoint; it must differ from start_location.
`num_eval_points`	`int`	No	`2`	Number of evaluation points, including endpoints; it must be at least 2.

2.5.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`name`	`str`	`None`	Unique, nonempty Probe name.
`region`	`ProbeRegion`	`None`	Portable target selection.
`variables`	`Tuple[str, ...]`	`()`	Output-variable names; the default is an empty tuple.
`start_location`	`Point3D`	`None`	First endpoint of the line segment.
`stop_location`	`Point3D`	`None`	Second endpoint; it must differ from start_location.
`num_eval_points`	`int`	`2`	Number of evaluation points, including endpoints; it must be at least 2.

Name	Type	Value	Description
`probeType`	`ProbeType`	`ProbeType.LINE`	Line-Probe type identifier.

2.5.b `to_dict`

Defined on: `coreform.probe.specs.LineProbeSpec.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.

2.5.c `to_json`

Defined on: `coreform.probe.specs.LineProbeSpec.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.

2.5.d `from_dict`

Defined on: `coreform.probe.specs.LineProbeSpec.from_dict`

Description: Create a validated specification from a mapping payload.

Call syntax

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

Returns

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

Signatures

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

Parameters

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

2.5.e `from_json`

Defined on: `coreform.probe.specs.LineProbeSpec.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

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

Returns

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

Signatures

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

Parameters

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

2.5.f `copy_with`

Defined on: `coreform.probe.specs.LineProbeSpec.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.

2.5.g `validate`

Defined on: `coreform.probe.specs.LineProbeSpec.validate`

Description: Return validation issues for this line-Probe specification.

Call syntax

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

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	Return validation issues for this line-Probe specification.

Signatures

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

Parameters

None.

2.6 `MultiPointProbeSpec`

Object path: `coreform.probe.specs.MultiPointProbeSpec`

Portable multi-point-Probe specification.

Call syntax

```
result = MultiPointProbeSpec(name=name, region=region, locations=locations)
```

Signatures

```
MultiPointProbeSpec(name: str = None, region: ProbeRegion = None, variables:
Tuple[str, ...] = tuple(), locations: Tuple[Point3D, ...] = tuple())
```

Parameters

Name	Type	Required	Default	Description
`name`	`str`	Yes	—	Unique, nonempty Probe name.
`region`	`ProbeRegion`	Yes	—	Portable target selection.
`variables`	`Tuple[str, ...]`	No	`()`	Output-variable names; the default is an empty tuple.
`locations`	`Tuple[Point3D, ...]`	Yes	—	One or more explicit evaluation locations.

2.6.a Attributes

Name	Type	Value	Description
`schemaVersion`	`int`	`0`	Version of the serialized specification schema.
`name`	`str`	`None`	Unique, nonempty Probe name.
`region`	`ProbeRegion`	`None`	Portable target selection.
`variables`	`Tuple[str, ...]`	`()`	Output-variable names; the default is an empty tuple.
`locations`	`Tuple[Point3D, ...]`	`field(default_factory=tuple, metadata={'required': True})`	One or more explicit evaluation locations.
`probeType`	`ProbeType`	`ProbeType.MULTI_POINT`	Multi-point-Probe type identifier.

2.6.b `validate`

Defined on: `coreform.probe.specs.MultiPointProbeSpec.validate`

Description: Return validation issues for this specification.

Call syntax

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

Returns

Type	Description
`Tuple[ValidationIssue, ...]`	All detected issues, or an empty tuple when the specification is valid.

Signatures

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

Parameters

None.

2.6.c `to_dict`

Defined on: `coreform.probe.specs.MultiPointProbeSpec.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.d `to_json`

Defined on: `coreform.probe.specs.MultiPointProbeSpec.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.e `from_dict`

Defined on: `coreform.probe.specs.MultiPointProbeSpec.from_dict`

Description: Create a validated specification from a mapping payload.

Call syntax

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

Returns

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

Signatures

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

Parameters

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

2.6.f `from_json`

Defined on: `coreform.probe.specs.MultiPointProbeSpec.from_json`

Description: Create a validated specification from a JSON document.

Call syntax

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

Returns

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

Signatures

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

Parameters

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

2.6.g `copy_with`

Defined on: `coreform.probe.specs.MultiPointProbeSpec.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 `ProbeSpec`

Object path: `coreform.probe.specs.ProbeSpec`

Type: `Union[PointProbeSpec, LineProbeSpec, MultiPointProbeSpec]`

Any supported portable Probe specification.