

# NANDRAD Modell-Referenz

Andreas Nicolai <https://github.com/ghorwin>, Dirk Weiß, Stephan Hirth, Katja  
Tribulowski, Hauke Hirsch, Anne Paepcke

Version 0.7 (05.02.2024)

# Inhaltsverzeichnis

1. Überblick	1
2. Struktur der Projektdatei	1
2.1. Eindeutigkeitsforderungen für Definitions-IDs	2
3. Grundlegende Datentypen in der NANDRAD-Projektdatei-Spezifikation	2
3.1. IBK:Parameter	2
3.2. IBK:IntPara	3
3.3. IBK:Flag	3
3.4. IBK:LinearSpline	4
3.5. LinearSplineParameter	4
4. Pfad-Platzhalter	5
5. Projektinformationen	6
6. Eingebettete Datenbanken	6
6.1. Materialien	7
6.2. Konstruktionstypen	8
6.2.1. Aktive/beheizte Schichten	9
6.3. Verglasungssysteme	9
7. Zonen	10
8. Konstruktionsinstanzen	12
8.1. Räumliche Diskretisierung (Finite-Volumen-Methode)	14
8.1.1. Algorithmus zur Erzeugung eines regulären Gitters	15
9. Interfaces (Konstruktions-Randbedingungen)	15
9.1. Wärmeleitung	16
9.2. Solare Absorption	17
9.3. Langwellige Emission	18
9.4. Dampfdiffusion	19
9.5. Luftstrom	20
10. Aktive Schichten/Flächenheizungen	21
11. Eingebettete Objekte (Fenster, Türen, Öffnungen)	21
11.1. Fenster	21
11.1.1. Fensterverschattung	22
12. Klimadaten und Standortinformationen	24
12.1. Übersicht	24
12.2. Spezifikation	24
12.2.1. Klimadateien	26
12.2.2. Gebäude-/Klimastandort	26
12.2.3. Zyklische (jährliche) und kontinuierliche (mehrjährige) Klimadaten	27
12.2.4. Zusätzliche Strahlungssensoren	29
12.3. Sonnenstrahlungsberechnung	31
12.4. Vorberechnete externe Verschattung/Eigenverschattung	31
12.4.1. Dateiformat für vorberechnete Sonnenlichtfaktoren	32
13. Objektlisten und Ergebnisreferenzen	33
13.1. Objektlisten-Definitionen	34

13.2. ID-Filter-Muster	35
14. Zeitpläne	35
14.1. Übersicht	35
14.2. Zeitplangruppen	37
14.3. Tagesschema-basierte Zeitpläne	37
14.3.1. Tägliche Zyklen	38
14.3.2. DailyCycle Zeitintervalle	41
14.3.3. Tägliche Zyklusparameterwerte	42
14.3.4. Vermeidung von Sprüngen / Leistungsverbesserung	43
14.4. Jahresschaltpläne	44
14.4.1. Definition von Jahreszeitplänen im XML-File	44
14.4.2. Definition von Jahreszeitplänen durch Einbindung von TSV-Dateien	45
14.5. Variablenliste	46
15. Outputs/Ergebnisse	47
15.1. Globale Ausgabeparameter	48
15.2. AusgaberaSTER	48
15.2.1. Regeln	49
15.3. Ausgangsdefinitionen	50
15.3.1. Variablennamen und Variablennachschlageregeln	51
15.3.2. Ausgabedateinamen	51
15.3.3. Zeittypen	52
15.3.4. Beispiele	53
15.4. Binäres Format	54
15.5. Solver-Logdateien	55
16. Globale Parameter	56
16.1. Simulationsparameter	56
16.1.1. Simulationszeitintervall	58
16.1.2. Simulationszeit und absoluter Zeitbezug	58
16.2. Solver-Parameter	59
16.2.1. Integrator	62
16.2.2. Linear equation system (LES) solver	63
16.2.3. Präkonditionierer	63
16.2.4. Solver-Fähigkeiten	63
17. Modellparametrisierung	64
17.1. Modellüberblick	64
17.2. Natürliches Lüftungsmodell	65
17.2.1. Regelbedingungen	70
17.2.2. Ausgabegrößen	70
17.3. Steuerungsmodell für Verschattung	71
17.3.1. Ausgabegrößen	72
17.4. Modell für interne Lasten	72
17.4.1. Ausgaben	74
17.5. Modell für interne Feuchtelasten	74
17.5.1. Ausgaben	75

17.6. Modell für Thermostate	75
17.6.1. TemperatureType	77
17.6.2. ControllerType	77
17.6.3. Ausgaben	78
17.7. Anlagensystem-Modell	78
17.8. Modell für ideale thermische Konditionierung	79
17.8.1. Heiz- und K�hleleistung	80
17.8.2. Ausgaben	80
17.9. Modell f�r ideale Fl�schenheizungen	81
17.9.1. Heiz- und K�hleleistung	82
17.9.2. Ausgaben	82
17.10. Modell f�r idealisierte Rohrregister-Fl�schenheizungen	82
17.10.1. Heiz- und K�hleleistung	85
17.10.2. Ausgaben	85
17.11. Modell f�r die Summation von Heiz- und K�hleleistungen	85
17.11.1. Ausgaben	86
17.12. Schnittstellen-Modell f�r die Anbindung externer Anlagennetze	86
17.12.1. Ausgaben	87
18. Thermo-hydraulische Netzwerke	87
18.1. Definition eines hydraulischen Netzwerks	88
18.1.1. Parameter	89
18.2. Medieneigenschaften	90
18.3. Rohreigenschaften	91
18.4. Komponentendefinitionen	92
18.4.1. Modelltyp: SimplePipe	93
18.4.2. Modelltyp: DynamicPipe	93
18.4.3. Modelltyp: ConstantPressurePump	93
18.4.4. Modelltyp: VariablePressurePump	95
18.4.5. Modelltyp: ConstantMassFluxPump	96
18.4.6. Modelltyp: ControlledPump	96
18.4.7. Modelltyp: ConstantPressureLossValve	98
18.4.8. Modelltyp: PressureLossElement	98
18.4.9. Modelltyp: ControlledValve	99
18.4.10. ModellTyp: HeatExchanger	99
18.4.11. ModellTyp: HeatPumpVariableIdealCarnotSourceSide	100
18.4.12. ModellTyp: HeatPumpVariableIdealCarnotSupplySide	101
ModellTyp: HeatPumpVariableSourceSide	101
18.4.13. Modelltyp: HeatPumpOnOffSourceSide	103
18.4.14. Modelltyp: IdealHeaterCooler	104
18.5. Str�mungselemente	105
18.5.1. Rohr-Elemente	107
18.6. W�rmeaustauschmodelle	108
18.6.1. Parameter f�r W�rmeaustauschdefinition	110
18.7. Regelung von Str�mungselementen	111

18.8. Regler und Massenstromkontrollmodelle .....	111
18.8.1. Regler: TemperatureDifference .....	112
18.8.2. Regler: TemperatureDifferenceOfFollowingElement .....	112
18.8.3. Regler: TemperatureDifferenceOfFollowingElement .....	113
18.8.4. Regler: ThermostatValue .....	113
18.8.5. Regler: MassFlux .....	113
18.8.6. Regler: PumpOperation .....	113
18.8.7. Regler: PressureDifferenceWorstpoint .....	113
18.8.8. Parameter f�r die Regler-Logik .....	114
18.9. Ausgaben .....	115
18.9.1. Verf�gbare Ausgaben .....	115
18.9.2. Ausgaben der rein hydraulischen Netzwerkberechnung .....	116
18.9.3. Ausgaben der thermo-hydraulischen Berechnung .....	116
18.9.4. Ausgaben des Netzwerks .....	117
19. Functional Mock-Up Interface .....	117
19.1. FMI-Variablendefinition .....	118
Appendix A: Einheitendefinitionen .....	118
Appendix B: Mengenreferenzen .....	121

# 1. †berblick

Dieses Dokument enthŠlt eine Beschreibung der verschiedenen Modelle und deren Parametrisierung in der NANDRAD Projektdatei. Dies ist primŠr eine Eingabereferenz.

Der Abschnitt [Struktur der Projektdatei](#) enthŠlt einen †berblick Ÿber die Struktur der Projektdatei mit Referenzen zu den einzelnen Abschnitten. Deshalb ist das ein guter Startpunkt, um sich einen †berblick Ÿber die NANDRAD Projektdefinition zu verschaffen.

## 2. Struktur der Projektdatei

Die NANDRAD-Projektspezifikation ist in einer XML-Datei mit der Erweiterung `nandrad` gespeichert. Der prinzipielle Aufbau der Datei sieht wie folgt aus:

Beispiel 1. Definition einer NANDRAD-Projektdatei

```
<?xml version="1.0" encoding="UTF-8" ?>
<NandradProject fileVersion="2.0">
  <!-- optional DirectoryPlaceholders section-->
  <DirectoryPlaceholders>...</DirectoryPlaceholders>

  <!-- the actual project specification -->
  <Project>
    <ProjectInfo>...</ProjectInfo>
    <Location>...</Location>
    <SimulationParameter>...</SimulationParameter>
    <SolverParameter>...</SolverParameter>
    <Zones>...</Zones>
    <ConstructionInstances>...</ConstructionInstances>
    <HydraulicNetworks>...</HydraulicNetworks>
    <ConstructionTypes>...</ConstructionTypes>
    <Materials>...</Materials>
    <Models>...</Models>
    <Schedules>...</Schedules>
    <Outputs>...</Outputs>
    <ObjectLists>...</ObjectLists>
    <!-- only needed for FMU export -->
    <FMI Description> ... </FMI Description>
  </Project>
</NandradProject>
```

Mit dem optionalen `DirectoryPlaceholders` kŠnnen relative Pfadplatzhalter definiert werden, die fŸr extern referenzierte Dateien verwendet werden sollen (siehe Abschnitt [Pfad-Platzhalter](#)).

Alle Projektdaten werden in den `<Project>`-tag eingeschlossen.

Eine Projektdatei kann die folgenden untergeordneten tags enthalten (die Reihenfolge ist beliebig):

XML-Tag	Beschreibung
<code>ProjectInfo</code>	Allgemeine Projekt-Meta-Daten ! <a href="#">Projektinformationen</a>
<code>Location</code>	Klimadaten und Standorteinstellungen ! <a href="#">Klimadaten und Standortinformationen</a>

XML-Tag	Beschreibung
<b>SimulationParameter</b>	Allgemeine Parameter des Simulationsmodells ! <a href="#">Simulationsparameter</a>
<b>SolverParameter</b>	Einstellungen der numerischen Algorithmen ! <a href="#">Solver-Parameter</a>
<b>Zones</b>	Zonenspezifikationen ! <a href="#">Zonen</a>
<b>ConstructionInstances</b>	Gebäudekomponenten und Randbedingungen ! <a href="#">Konstruktionsinstanzen</a>
<b>HydraulicNetworks</b>	Thermohydraulische Netze ! <a href="#">Thermo-hydraulische Netzwerke</a>
<b>ConstructionTypes</b>	Datenbank von mehrschichtigen Konstruktionen ! <a href="#">Konstruktionstypen</a>
<b>Materials</b>	Materialdatenbank ! <a href="#">Materialien</a>
<b>Models</b>	Modell-Parameterblöcke ! <a href="#">Modellparametrisierung</a>
<b>Schedules</b>	Definition von geplanten Parametern ! <a href="#">Zeitpläne</a>
<b>Outputs</b>	Definition von Ausgaben ! <a href="#">Outputs/Ergebnisse</a>
<b>ObjectLists</b>	Definition von Objektlisten/Objektreferenzgruppe ! <a href="#">Objektlisten und Ergebnisreferenzen</a>
<b>FMIDescription</b>	Definition der FMU Export-Schnittstelle ! <a href="#">Functional Mock-Up Interface</a>

## 2.1. Eindeutigkeitsforderungen für Definitions-IDs

Im NANDRAD müssen folgenden Objekttypen in einem einzigen ID-Raum definiert werden, d.h. die IDs der unterschiedlichen Objekte dürfen sich nicht doppeln:

- ✖ **Zone**
- ✖ **ConstructionInstance**
- ✖ **EmbeddedObjects**
- ✖ **Location.Sensors**

## 3. Grundlegende Datentypen in der NANDRAD-Projektdatei-Spezifikation

Innerhalb der verschiedenen Spezifikationsabschnitte der Projektdatei werden einige grundlegende Datentypen / XML-tags häufig verwendet. Die Regeln für die Spezifikation dieser Parameter sind im Folgenden definiert.

### 3.1. IBK:Parameter

Ein XML-Tag mit dem Namen **IBK:Parameter** definiert einen Fließkommaparameter (floating point value

parameter), der durch einen Namen und eine physikalische Einheit identifiziert wird (obligatorische XML-Attribute `name` und `unit`). Der Wert des XML-tags ist der eigentliche Parameterwert.

#### Beispiel 2. Parameter mit verschiedenen Einheiten

```
<IBK:Parameter name="Volume" unit="m3">30</IBK:Parameter>
<IBK:Parameter name="Temperature" unit="C">20</IBK:Parameter>
<IBK:Parameter name="Temperature" unit="K">293.15</IBK:Parameter>
<!-- unitless parameters take the --- unit --->
<IBK:Parameter name="Rel Tol" unit="---">0.7</IBK:Parameter>
```

Die Einheiten müssen aus der globalen Einheitenliste ausgewählt werden, siehe Abschnitt [Einheitendefinitionen](#). Wird ein Parameter nicht definiert, wird er als *fehlend* markiert, was bedeutet, dass entweder ein Standardwert verwendet wird oder - im Falle von obligatorischen Benutzerparametern - ein Fehler ausgelöst wird.

### 3.2. IBK:IntPara

Dieser Parameter wird für Flags verwendet. Das obligatorische Attribut `name` identifiziert das Flag. Wird ein Flag nicht definiert, wird es als *fehlend* markiert, was bedeutet, dass entweder ein Standardwert verwendet wird oder - im Falle von obligatorischen Benutzerparametern - ein Fehler ausgelöst wird.

Wird für ganzzahlige Parameter verwendet. Das obligatorische Attribut `Name` identifiziert den Parameter. Der XML-tag `value` ist der Parameterwert. Wird ein Parameter nicht definiert, wird er als *fehlend* markiert, was bedeutet, dass entweder ein Standardwert verwendet wird oder - im Falle von obligatorischen Benutzerparametern - ein Fehler ausgelöst wird.

#### Beispiel 3. Ganze Zahl (Integer) Parameter-Definition

```
<IBK: IntPara name="DiscMaxElementsPerLayer">30</IBK: IntPara>
```

### 3.3. IBK:Flag

Wird für boolische Schalter (An/Aus-Optionen) verwendet. Das obligatorische Attribut `name` identifiziert das Flag. Wird ein Flag nicht definiert, wird es als *fehlend* markiert, was bedeutet, dass entweder ein Standardwert verwendet wird oder - im Falle von obligatorischen Benutzerparametern - ein Fehler ausgelöst wird.

#### Beispiel 4. Flag Definition

```
<IBK: Flag name="EnableCyclicSchedules">true</IBK: Flag>
```



Erkannte Werte für Flag-Parameter sind **true** und **1** oder **false** und **0**.

### 3.4. IBK:LinearSpline

Eine lineare Spline ist effektiv eine Datentabelle mit x- und y-Werten, wobei die x-Werte streng monoton steigende Werte sind. Das obligatorische Attribut **name** identifiziert die lineare Spline. Die untergeordneten Tags **X** und **Y** enthalten die tatsächlichen Werte, immer ohne Einheit. Die Anzahl der x- und y-Werte muss übereinstimmen.

#### Beispiel 5. Lineare Spline-Definition

```
<IBK:LinearSpline name="Thermal Load">
  <X unit="-">0 6 8 10 17 18 19 20</X>
  <Y unit="-">0 0.5 0.8 1.0 0.7 0.6 0.5 0</Y>
</IBK:LinearSpline>
```

### 3.5. LinearSplineParameter

Ein LinearSpline-Parameter ist effektiv ein erweiterter **IBK:LinearSpline**-Parameter mit zusätzlichen Attributen.

#### Beispiel 6. LinearSplineParameter Definition

```
<LinearSplineParameter name="Thermal Load" interpolationMethod="linear">
  <X unit="h">0 6 8 10 17 18 19 20</X>
  <Y unit="W">0 0.5 0.8 1.0 0.7 0.6 0.5 0</Y>
</LinearSplineParameter>
```

Tabelle 1. Attribute

Attribut	Beschreibung	Format	Verwendung
<b>name</b>	Spezifischer Name, der sich auf den Raumtyp bezieht, für den der Jahresplan gesetzt wird	string	<i>required</i>
<b>interpolationMethod</b>	Gibt die Interpolationsmethode zwischen den definierten y-Werten an.  ¥ <b>constant</b> - konstante Interpolation (Werte konstant während des Zeitschritts)  ¥ <b>linear</b> - lineare Interpolation (Werte linear interpoliert zwischen Zeitschritten)	Schlüsselwort	<i>optional</i> ( <b>linear</b> ist der Standard)

Attribut	Beschreibung	Format	Verwendung
<code>wrapMethod</code>	<p>Gibt an, was getan werden soll, wenn Werte mit x-Werten außerhalb des x-Wertebereichs angefordert werden.</p> <p>¥ <code>continuous</code> - konstante Extrapolation (ersten bzw. letzten Wert nehmen)</p> <p>¥ <code>cyclic</code> - zyklische Anpassung mit der modellspezifischen Periodenlänge anwenden (z. B. ein Jahr)</p>	key	<i>optional</i> ( <code>continuous</code> ist der Standard)

Die Child-Tags `X` und `Y` enthalten jeweils ein obligatorisches Attribut `unit` mit der jeweiligen Werteinheit (siehe [Einheitendefinitionen](#)).

Alternativ kann man auch eine Datei mit Tabulator-getrennten Spalten angeben, unter Verwendung des XML-tags `TSVFile`.

#### Beispiel 7. Linear Spline-Definition mit Angabe der Datei

```
<LinearSplineParameter name="HeatExchangeSpline" interpolationMethod="linear">
  <TSVFile>${Project Directory}/climate/Temperature.csv?3</TSVFile>
</LinearSplineParameter>
```

#### Beispiel 8. Dazugehörige Datei `Temperature.csv`

```
Time [h] Temp [C] otherTemp [C] anotherTemp [C]
0 0 0 0
12 5 7 -9
36 -8 12 65
```

Eine Datei im tsv-Format enthält in der ersten Spalte Zeitwerte und haben danach eine beliebige Anzahl von Datenspalten. Gibt es mehr als eine Datenspalte, muss die Auswahl der Datenspalte durch Anhang des Spezifizierers `?<colIndex>` erfolgen. Die erste Datenspalte hat den Index 1. Daher bezeichnet `?3` wie im Beispiel oben die dritte Spalte (`anotherTemp` im Beispiel oben).



Es ist möglich, Pfad-Platzhalter im Dateinamen zu verwenden (siehe [Pfad-Platzhalter](#)).



Man kann entweder `X` UND `Y` angeben, oder alternativ `TSVFile`. Beides ist nicht erlaubt und führt zu einem Fehler.

## 4. Pfad-Platzhalter

In einigen Teilen der NANDRAD-Projektdatei werden externe Dateien referenziert (z.B. Klimadaten-Dateien, siehe [Klimadateien](#)). Um den Austausch von Projekten oder Referenzdatendateien in

gemeinsamen Datenbankverzeichnissen zu vereinfachen, ist es möglich, Pfadplatzhalter in Dateipfaden zu verwenden.

Sie können z. B. `${MyDatabase}` als `/home/sim/climate_DB` definieren und dann in Ihrem Projekt eine Klimadatendatei referenzieren über `${MyDatabase}/ClimateData.epw`.

Diese Zuordnung der Platzhalter wird zu Beginn der Projektdatei vorgenommen, sodass beim Austausch von Projektdateien zwischen Computern die Platzhalterpfade zu den Verzeichnissen auf dem lokalen Rechner leicht geändert werden können, ohne dass weitere Änderungen in der Projektdatei erforderlich sind.

Die einzelnen Pfadplatzhalter werden in den `DirectoryPlaceholders` definiert:

#### Beispiel 9. Benutzerdefinierte Directory Placeholders

```
<DirectoryPlaceholders>
  <Placeholder name="Climate_DB">/home/sim/climate_DB</Placeholder>
  <Placeholder name="DataFiles">/home/sim/data</Placeholder>
</DirectoryPlaceholders>
```

Es gibt einen eingebauten Platzhalter `${Project Directory}`, der automatisch mit dem Pfad zum Verzeichnis der Projektdatei definiert wird.

## 5. Projektinformationen

Dieser Abschnitt enthält Änderungszeiten/-daten und eine kurze Beschreibung des Projekts. Die folgenden untergeordneten tags werden unterstützt.

Child-Tag	Beschreibung	Format
<code>Comment</code>	Allgemeiner Kommentar zum Projekt.	<i>string</i>
<code>Created</code>	Datum/Uhrzeit der Erstellung dieses Projektes.	<i>string</i>
<code>LastEdited</code>	Datum/Uhrzeit der letzten Änderung des Projektes.	<i>string</i>

Die Datum/Uhrzeit-Strings für `Created` und `LastEdited` sollten das Datum und die Uhrzeit in einem für den Benutzer lesbaren Format speichern, da sie zum Anzeigen von Listen der Projekte mit Änderungs-/Erstellungsdatum verwendet werden können.

## 6. Eingebettete Datenbanken

Um Gebäudekomponenten wie Wände, Decken und Böden usw. zu modellieren, ist es notwendig, einige Parameter für die Materialien zu definieren. Damit ist es dann möglich Konstruktionen zu definieren, die aus solchen Materialien bestehen. Diese Parameter werden in Datenbanken gespeichert, die eigentlich Listen aus XML-Objekten sind.

# 6.1. Materialien

In der NANDRAD-Projektdatei beginnt der Abschnitt der Materialdatenbank mit einem XML-tag namens **Materials**.

Beispiel 10. Materialien mit Parametern

```
<Materials>
  <Material id="1001" displayName="Backstein">
    <IBK:Parameter name="Density" unit="kg/m3">2000</IBK:Parameter>
    <IBK:Parameter name="HeatCapacity" unit="J/kgK">1000</IBK:Parameter>
    <IBK:Parameter name="Conductivity" unit="W/mK">1,2</IBK:Parameter>
  </Material>
  <Material id="1004" displayName="Gute Dämmung">
    <IBK:Parameter name="Density" unit="kg/m3">50</IBK:Parameter>
    <IBK:Parameter name="HeatCapacity" unit="J/kgK">1000</IBK:Parameter>
    <IBK:Parameter name="Conductivity" unit="W/mK">0,02</IBK:Parameter>
  </Material>
</Materials>
```

In diesem tag beginnt jedes Materialeigenschaften-Set mit einem XML-tag namens **Material** mit zwei XML-Attributen **id** und **displayName**.

Tabelle 2. Attribute

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Eindeutige ID des Materials.	> 0	<i>erforderlich</i>
<b>displayName</b>	Name des Materials (wird für Informations-/Fehlermeldungen verwendet)	string	<i>optional</i>

Für die Materialparameter wie Dichte, Wärmekapazität und Wärmeleitfähigkeit müssen diese im XML-tag **IBK:Parameter** definiert werden (siehe [IBK:Parameter](#)):

Name	Standardeinheit	Beschreibung	Wertebereich	Verwendung
<b>Density</b>	kg/m3	Trockendichte des Materials.	> 0,01	<i>erforderlich</i>
<b>HeatCapacity</b>	J/kgK	Spezifische Wärmekapazität des Materials.	>= 100	<i>erforderlich</i>
<b>Conductivity</b>	W/mK	Wärmeleitfähigkeit des trockenen Materials.	>= 1e-5	<i>erforderlich</i>

## 6.2. Konstruktionstypen

Konstruktionen werden innerhalb des Abschnitts definiert, der mit einem XML-tag `ConstructionTypes` beginnt.

*Beispiel 11. construction types mit Referenzen zu Materialobjekten*

```
<ConstructionTypes>
  <ConstructionTypes id="10005" displayName="Testkonstruktion">
    <MaterialLayers>
      <MaterialLayer thickness="0.2" matId="1001" /> <!-- room side -->
      <MaterialLayer thickness="0.3" matId="1004" />
    </MaterialLayers>
  </ConstructionTypes>
</ConstructionTypes>
```

Innerhalb dieses Abschnitts beginnt jede Konstruktionsdefinition mit dem XML-tag `ConstructionType` mit den XML-Attributen `id` und optional `displayName`:

*Tabelle 3. Attribute*

Attribut	Beschreibung	Format	Verwendung
<code>id</code>	Eindeutige Identifikationsnummer	positive Ganzzahl ( $> 0$ )	<i>erforderlich</i>
<code>displayName</code>	Name der Konstruktion (wird für Informations-/Fehlermeldungen verwendet)	string	<i>optional</i>

Eine Konstruktion besteht aus einer oder mehreren Materialschichten. Diese werden im untergeordneten XML-tag mit dem Namen `MaterialLayers` definiert. Jede Materialschicht wird mit dem XML-tag `MaterialLayer` mit den folgenden XML-Attributen definiert:

XML-Attribut	Beschreibung	Format	Verwendung
<code>thickness</code>	definiert die Dicke der Schicht in <code>m</code>	$> 0.0$	<i>erforderlich</i>
<code>matId</code>	verweist auf ein Material durch eine eindeutige Material-Identifikationsnummer ( <code>id</code> wie in einem <code>Material</code> -Tag definiert)	string	<i>erforderlich</i>

Das Material in der Materialschicht wird über das Attribut `matId` mittels ID referenziert. Das `MaterialLayer` hat keine Child-tags, da alle benötigten Daten wie oben beschrieben als XML-Attribute definiert sind.

### 6.2.1. Aktive/beheizte Schichten

Jeder Konstruktionsaufbau kann genau eine aktive Schicht haben, welche mit einem Heizregister versehen wird. Falls eine solche Schicht existiert, muss das XML-tag **ActiveLayerIndex** angegeben sein. Dieses XML-Element enthält den 0-basierten Index der Schicht, d.h. Index 0 entspricht der ersten Schicht in der **Material Layers**-Liste.

Falls eine aktive Schicht definiert wurde, muss es irgendwo ein Modell geben, welches passend dafür eine Heiz-/Kühlleistung berechnet. Beispielsweise kann dies eine Fußbodenheizung sein (siehe **Modell für ideale Fliesenheizungen** oder **Modell für idealisierte Rohrregister-Fliesenheizungen**).

## 6.3. Verglasungssysteme

Verglasungssysteme werden in einer Liste innerhalb des XML-tags **WindowGlazingSystems** definiert.

*Beispiel 12. Parameterdefinition für ein Verglasungssystem*

```
<WindowGlazingSystems>
  <WindowGlazingSystem id="123" modelType="Simple">
    <IBK:Parameter name="ThermalTransmittance" unit="W/m2K">0,4</IBK:Parameter>
    <LinearSplineParameter name="SHGC" interpolationMethod="linear" wrapMethod="cyclic">
      <!-- X incidence angle - 90 deg = Sonne steht senkrecht/normal zur Oberfläche -->
      <X unit="Deg">0 90 </X>
      <!-- Hinweis: kein konstanter Parameter - SHGC konstant wird wie unten definiert -->
      <Y unit="---">0.6 0.6 </Y>
    </LinearSplineParameter>
  </WindowGlazingSystem>
</WindowGlazingSystems>
```

Innerhalb dieses Abschnitts beginnt jede Definition eines Verglasungssystem mit dem XML-tag **WindowGlazingSystem** mit den XML-Attributen **id**, **modelType** und optional **displayName**:

*Tabelle 4. Attribute*

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Eindeutige Identifikationsnummer	positive integer ( > 0 )	<i>erforderlich</i>
<b>displayName</b>	Name des Verglasungssystems (wird für Informations-/Fehlermeldungen verwendet)	string	<i>optional</i>
<b>modelType</b>	Identifiziert die Modellkomplexität:  ‣ <b>Simple</b> - Standard-Verglasungsmodell, mit einem U-Wert (Wärmedurchgangskoeffizient) und einfallswinkelabhängigem SHGC-Wert	string	<i>erforderlich</i>

Skalare Parameter werden innerhalb eines XML-tags **IBK:Parameter** definiert (siehe **IBK:Parameter**):

Name	Standardeinheit	Beschreibung	Wertebereich	Verwendung
Thermal Transmittance	W/m <sup>2</sup> K	Wärmedurchgangskoeffizient der Verglasung	> 0	erforderlich für Modelltyp Simple

Parameter, die vom Einfallswinkel abhängen, werden in einem XML-tag `LinearSplineParameter` definiert (siehe [LinearSplineParameter](#)):

Name	Standardeinheit	Beschreibung	Wertebereich	Verwendung
SHGC	---	Solarer Wärmeenergiewirkungskoeffizient	> 0	erforderlich für Modelltyp Simple

## 7. Zonen

Um Gebäude modellieren zu können, ist es notwendig die einzelnen Räume mit den entsprechenden Parametern zu definieren. Eine Zone definiert einen thermisch gut durchmischten Bereich/Raum mit einer einzigen/einheitlichen Lufttemperatur.

Objekte vom Typ `Zone` speichern alle Eigenschaften die benötigt werden, um die Zonentemperatur aus der Energiedichte (der Erhaltungsgroße) zu berechnen.

### Beispiel 13. Definition der Zonen

```
<Zones>
  <Zone id="1" displayName="Var01" type="Active">
    <IBK:Parameter name="Area" unit="m2">10</IBK:Parameter>
    <IBK:Parameter name="Volume" unit="m3">30</IBK:Parameter>
  </Zone>
</Zones>
```

Innerhalb des XML-tags namens `Zones` beginnt jede Zone mit dem XML-tag `Zone`. Die folgenden XML-Attribute müssen definiert werden:

```
<Zone id="1" displayName="Var01" type="Active">
```

### Tabelle 5. Attribute

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	eindeutige Identifikationsnummer der Zone	(1-100)	<i>erforderlich</i>
<b>displayName</b>	Anzeigename der Zone. Wird benötigt, um die Zone im Datenmodell und in Ausgaben leichter zu finden.	string	<i>optional</i>
<b>type</b>	<p>Legt fest, ob die Zone ausgeglichen und in das Gleichungssystem einbezogen wird.</p> <p>           ☒ <b>Constant</b> als Zone mit konstanten Temperaturen (Parameter)            ☒ <b>Scheduled</b> als Zone mit zeitplandefinierten Temperaturen            ☒ <b>Active</b> als Zone, die durch einen Temperaturknoten im Raum beschrieben wird            ☒ <b>Ground</b> als Bodenzone (berechnet die Temperatur auf Basis des Standards)         </p>	key	<i>erforderlich</i>

F  r *konstante* Zonen wird angenommen, dass die Temperatur vorgegeben ist, w  hrend in *Active* Zonen die Temperatur berechnet wird (d. h. in den Unbekannten des Modells enthalten ist).

Eine `_konstante_` Zone ben  tigt nur den Temperaturparameter.

Eine `_scheduled_` Zone ben  tigt keine Parameter. Der Temperaturzeitplan muss allerdings in den `_Schedules_` abgelegt sein, eine `_ScheduleGroup_` f  r eine die Zone enthaltene Objektliste definieren und den Parameter `'TemperatureSchedule'` definieren.

Parameter (siehe Abschnitt [IBK:Parameter](#) f  r eine Beschreibung des tags [IBK:Parameter](#)):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<b>Volume</b>	m3	Zonenluftvolumen	[0.0-10.0]	<i>erforderlich</i>
<b>Area</b>	m2	Nettonutzfl��che des Erdgeschosses (f��r fl��chenbezogene Leistungen und Lasten)	[0.0-10.0]	<i>optional</i>
<b>HeatCapacity</b>	J/K	Zus��tzliche W��rmekapazit��t (M��bel, etc.)	[0.0-10.0]	<i>optional</i>
<b>Temperature</b>	C	Temperatur der Zone  nur verwendet, wenn <b>constant</b>	-70��120	<i>(erforderlich)</i>



Name	Einheit	Beschreibung	Wertebereich	Verwendung
RelativeHumidity	%	Relative Luftfeuchtigkeit der Zone nur verwendet, wenn constant	0-100	(erforderlich)
CO2Concentration	g/m3	CO2-Konzentration der Zone nur verwendet, wenn constant	0-1000	(erforderlich)



Die Parameter **RelativeHumidity** und **CO2Concentration** müssen nur für *konstante* Zonen definiert werden, wenn die jeweilige Bilanzgleichung aktiviert ist.

## 8. Konstruktionsinstanzen

Konstruktionsinstanzen repräsentieren tatsächlich verbaute eindimensionale Teile der Gebäudehülle, z.B. Wände, Böden, Decken, Türen.

*Beispiel 14. Definition einer Außenwand nur mit der Randbedingung Wärmeleitung*

```

<ConstructionInstances>
  <!-- Oberfläche Var 01 -->
  <ConstructionInstances id="1" displayName="All Surfaces Var01">
    <ConstructionTypeId>10005</ConstructionTypeId>
    <IBK:Parameter name="Area" unit="m2">62</IBK:Parameter>
    <InterfaceA id="10" zoneId="1">
      <!--Interface zu 'Room'-->
      <InterfaceHeatConduction modelType="Constant">
        <IBK:Parameter name="HeatTransferCoefficient" unit="W/m2K">2.5</IBK:Parameter>
      </InterfaceHeatConduction>
    </InterfaceA>
    <InterfaceB id="11" zoneId="0">
      <!--Schnittstelle nach außen-->
      <InterfaceHeatConduction modelType="Constant">
        <IBK:Parameter name="HeatTransferCoefficient" unit="W/m2K">8</IBK:Parameter>
      </InterfaceHeatConduction>
    </InterfaceB>
  </ConstructionInstances>
</ConstructionInstances>

```

Die Konstruktionsinstanzen werden innerhalb des XML-tags **ConstructionInstances** definiert. Innerhalb des Abschnitts beginnt jede Konstruktionsdefinition mit dem XML-tag **ConstructionInstance** mit den Attributen **id** und **displayName**.

*Tabelle 6. Attribute*

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Identifikationsnummer der Konstruktionsinstanz	int	<i>erforderlich</i>
<b>displayName</b>	Anzeigename der Konstruktionsinstanz. Wird benötigt, um die Konstruktionsinstanz im Datenmodell und in der Ausgaben leichter zu finden.	string	<i>optional</i>

Die Konstruktionsinstanz hat das folgende *erforderliche* Child-tags:

Tabelle 7. Construction Instance Child tags

Tag	Beschreibung
<b>ConstructionTypeId</b>	Referenz auf <b>ConstructionType</b>
<b>IBK:Parameter</b>	Verschiedene <b>IBK:Parameter</b> für Konstruktionsinstanz
<b>InterfaceA</b>	Schnittstelle für Konstruktionsinstanz Seite A
<b>InterfaceB</b>	Schnittstelle für Konstruktionsinstanz Seite B

**ConstructionTypeId** ist eindeutige Id, die den Konstruktionstyp (Schichtenaufbau) der Konstruktionsinstanz definiert (siehe [Konstruktionstypen](#)).

Für die Parameter der Konstruktionsinstanz können die folgenden XML-tags mit dem Namen **IBK:Parameters** mit den XML-Attributen **name** und **unit** mit den folgenden Einträgen definiert werden:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<b>Orientation</b>	Deg	Ausrichtung der Wand  wenn eine Schnittstelle eine solare (kurzwellige) Strahlungs-Randbedingung hat, ist sie <i>erforderlich</i>	0°-360	<i>erforderlich / optional</i>
<b>Inclination</b>	Deg	Neigung der Wand  0° - Dach 90 Grad - senkrechte Wand 180 Deg - nach unten gerichtet  wenn eine Schnittstelle kurz- und/oder langwellige Strahlungsrandbedingung hat, ist sie <i>erforderlich</i>	0°-180	<i>erforderlich / optional</i>
<b>Area</b>	m <sup>2</sup>	Bruttofläche der Wand (inkl. evtl. vorhandener Fenster, Löcher etc.)	> 0	<i>erforderlich</i>

Darin müssen die Schnittstellen mit dem XML-tag `InterfaceA` und `InterfaceB` angegeben werden. Schließlich müssen die Interfaces mit dem XML-tag `InterfaceA` und `InterfaceB` mit den XML-Attributen `id` und `zoneId` definiert werden. Im Folgenden wird dies im Detail beschrieben.

## 8.1. Räumliche Diskretisierung (Finite-Volumen-Methode)

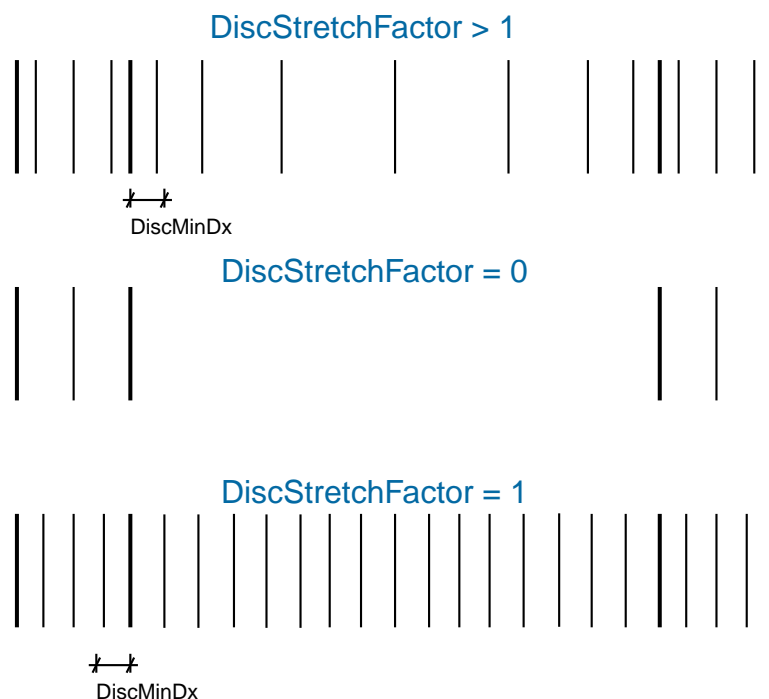
Während der Berechnung wird jede der Konstruktionen mit Hilfe eines Algorithmus zur Gittergenerierung räumlich diskretisiert. Dieser Algorithmus verwendet drei einflussreiche Parameter, die im Abschnitt [Solver-Parameter](#) definiert sind:

• `DiscMinDx`

• `DiscStretchFactor`

• `DiscMaxElementsPerLayer`

[Abbildung 1](#) veranschaulicht die Wirkung verschiedener Dehnungsfaktoren



*Abbildung 1. Verschiedene Diskretisierungsvarianten in Abhängigkeit vom Parameter `DiscStretchFactor`*

Grundsätzlich werden drei verschiedene Gittergenerierungsverfahren unterstützt:

• **minimal grid:** bei `DiscStretchFactor = 0` erzeugt der Algorithmus ein Finite Volumen pro Materialschicht, mit Ausnahme der Randelemente, die immer in zwei aufgeteilt werden (notwendig für die Oberflächenwertextrapolation). So ergeben sich z. B. bei einem 4-Schicht-Aufbau 6 Finite Volumen.

- ¥ equidistant: bei `DiscStretchFactor = 1` erzeugt der Algorithmus in jeder Schicht gleichmäßig verteilte Gitterelemente, deren Dicke nahe, aber immer kleiner als der Parameter `DiscMinDx` ist. Da Materialschichten unterschiedliche Breiten haben können, ist eine einheitliche Dicke der Gitterelemente in der gesamten Konstruktion möglicherweise nicht möglich. Wählen Sie einen `DiscMinDx`-Parameter, bei dem alle Materialschichtbreiten ganzzahlige Vielfache dieser Rasterelementdicke sind (z.B. `1 mm`)
- ¥ regular grid: für jeden `DiscStretchFactor > 1` wird ein regelmäßiges, variabel beabstandetes Gitter erzeugt.

### 8.1.1. Algorithmus zur Erzeugung eines regulären Gitters

Ein regelmäßiges Streckgitter wird mit einer doppelseitigen *tanh*-Streckfunktion erzeugt. Der Faktor `DiscStretchFactor` bestimmt dabei ungefähr das Verhältnis der ersten beiden Gitterelementbreiten. Natürlich variiert dieser Wachstumsfaktor und geht in der Mitte einer Materialschicht gegen Null, aber er bestimmt sehr schön den gesamten Gitterausschnitt. Ein Faktor von 4 ist ein guter Standardwert.

Der Parameter `DiscMinDx` definiert die maximale Breite der äußersten Gitterelemente in jeder Schicht. Damit wird indirekt auch die Anzahl der Gitterelemente pro Materialschicht bestimmt. Mit zunehmender Anzahl von Gitterelementen pro Schicht werden die äußersten Gitterelemente kleiner. Auf diese Weise bestimmt der Algorithmus die Anzahl der Gitterzellen (für einen gegebenen `DiscStretchFactor`), bis die erzeugte Breite bei den äußersten Gitterelementen gleich oder kleiner als der Parameter `DiscMinDx` ist. Eine minimale Elementdicke von `2 mm` ist ein guter Standardwert für sehr genaue Berechnungen, aber ein Wert von `5 mm` kann in vielen Situationen ausreichen (dies reduziert die Anzahl der Unbekannten und eventuell die Simulationszeit erheblich).

Schließlich gibt es noch den Parameter `DiscMaxElementsPerLayer`, mit dem die Anzahl der zu erzeugenden Gitterelemente in einer Materialschicht begrenzt werden kann. Dies ist besonders dann sinnvoll, wenn sehr dicke Materialschichten vorhanden sind und eine große Anzahl von Gitterzellen erzeugt wird. Oft wird diese Genauigkeit nicht benötigt (jedenfalls bei sehr dicken Materialschichten), so dass eine Begrenzung der Anzahl zur Beschleunigung der Berechnung sinnvoll sein kann. Solange die Anzahl der erzeugten Gitterzellen pro Materialschicht `DiscMaxElementsPerLayer` überschreitet, wird der Algorithmus den `DiscStretchFactor` schrittweise erhöhen, bis das Kriterium erfüllt ist. Der Solver wird für jede Konstruktionsschicht, auf die diese Anpassung angewendet wird, eine Warnmeldung ausgeben.

!

Wie bei allen numerischen Lösern, die mit Rechengittern arbeiten, gibt es immer einen Kompromiss zwischen Geschwindigkeit und Genauigkeit. Eine Studie über die Empfindlichkeit des Gitters kann hilfreich sein, z. B. indem Sie mit `DiscMinDx = 5 mm` und `DiscStretchFactor = 8` beginnen und dann die Werte schrittweise reduzieren, bis sich die Lösung nicht mehr verändert. Für kleine Gebäude/Modelle, bei denen die Leistung keine Rolle spielt, können die Standardwerte `DiscMinDx = 2 mm` und `DiscStretchFactor = 4` verwendet werden.

## 9. Interfaces (Konstruktions-Randbedingungen)

Die Interfaces definieren Randbedingungen und Parameter für die ein oder zwei Oberflächen `InterfaceA` und `InterfaceB` einer Konstruktionsinstanz. Wenn die Konstruktionsinstanz eine adiabatische Wand

definiert, wird nur ein Interface benötigt. In allen anderen Fällen werden zwei Schnittstellen benötigt. Das **InterfaceA** verknüpft die erste Materialschicht aus dem Konstruktionstyp mit der zugeordneten Zone über die **zoneId**. Das **InterfaceB** verknüpft die letzte Materialschicht aus dem Konstruktionstyp mit der **zoneId** von **InterfaceB**.

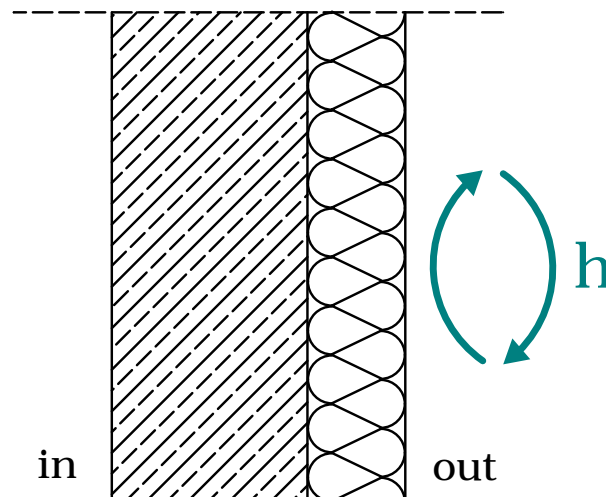
*Beispiel 15. Schnittstellendefinitionen für eine Konstruktion mit Schnittstellen für beide Seiten*

```
<ConstructionInstance id="1" displayName="All Surfaces Var01">
  ...
  <InterfaceA id="10" zoneId="1">
    <InterfaceHeatConduction modelType="Constant">
      <IBK:Parameter name="HeatTransferCoefficient" unit="W/m2K">2.5</IBK:Parameter>
    </InterfaceHeatConduction>
  </InterfaceA>
  <InterfaceB id="11" zoneId="0">
    <InterfaceHeatConduction modelType="Constant">
      <IBK:Parameter name="HeatTransferCoefficient" unit="W/m2K">8</IBK:Parameter>
    </InterfaceHeatConduction>
    <InterfaceSolarAbsorption model="Constant">
      <IBK:Parameter name="AbsorptionCoefficient" unit="---">0.6</IBK:Parameter>
    </InterfaceSolarAbsorption>
    <InterfaceLongWaveEmission model="Constant">
      <IBK:Parameter name="Emissivity" unit="---">0.9</IBK:Parameter>
    </InterfaceLongWaveEmission>
  </InterfaceB>
</ConstructionInstance>
```

**InterfaceA** und **InterfaceB** können ein oder mehrere untergeordnete tags haben.

## 9.1. Wärmeleitung

Die konvektive Wärmeleitung über die Schnittstelle wird durch das XML-tag **InterfaceHeatConduction** beschrieben.



Beispiel 16. Parameterdefinition für die Randbedingung Wärmeleitung

```
<InterfaceHeatConduction modelType="Constant">
  <IBK:Parameter name="HeatTransferCoefficient" unit="W/m2K">2.5</IBK:Parameter>
</InterfaceHeatConduction>
```

Die `InterfaceHeatConduction` muss mit dem folgenden XML-Attribut `modelType` definiert werden.

Tabelle 8. Attribute

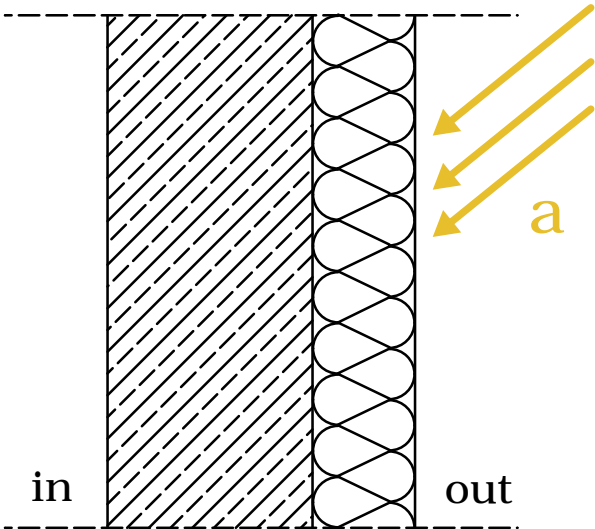
Attribut	Beschreibung	Format	Verwendung
<code>modelType</code>	Setzt den Typ des Wärmeleitungsmodells  ☒ <code>Constant</code> - es wird ein konstantes Modell verwendet (derzeit die einzige Option)	key	erforderlich

Fließkommaparameter (siehe Abschnitt `IBK:Parameter` für eine Beschreibung des tags `IBK:Parameter`):

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
<code>HeatTransferCoefficient</code>	W/m2K	Konstanter konvektiver Wärmeübergangskoeffizient	$\in \mathbb{R}_{>0.0}$	erforderlich

9.2. Solare Absorption

Die solare Absorption über die Schnittstelle wird durch das XML-tag `InterfaceSolarAbsorption` beschrieben. Dieser Koeffizient beschreibt die solare Kurzwellenstrahlung, die von der Grenzfläche absorbiert wird.



### Beispiel 17. Parameterdefinition für die Randbedingung Solare Absorption

```
<InterfaceSolarAbsorption modelType="Constant">
  <IBK:Parameter name="AbsorptionCoefficient" unit="---">0.6</IBK:Parameter>
</InterfaceHeatConduction>
```

Das `InterfaceSolarAbsorption` muss mit dem folgenden XML-Attribut `modelType` definiert werden.

Tabelle 9. Attribute

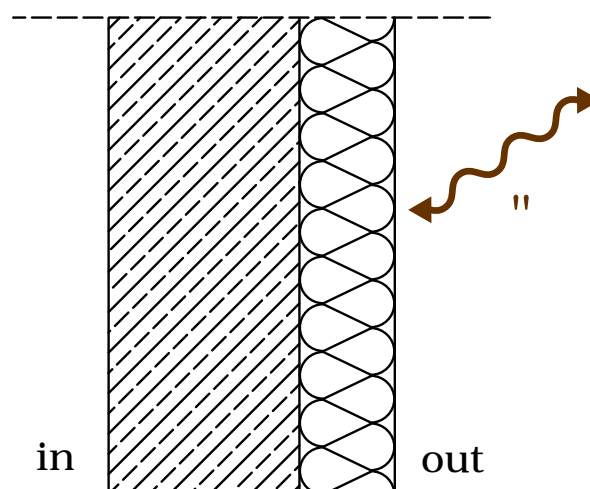
Attribut	Beschreibung	Format	Verwendung
<code>modelType</code>	Setzt den Typ des Wärmeleitungsmodells  $\forall$ <code>Constant</code> - es wird ein konstantes Modell verwendet (derzeit die einzige Option)	key	erforderlich

Es können XML-tags mit dem Namen `IBK:Parameter` mit den XML-Attributen `name` und `unit` mit den folgenden Einträgen definiert werden:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>AbsorptionCoefficient</code>	---	Konstanter Absorptionskoeffizient	0 ÷ 1	erforderlich

## 9.3. Langwellige Emission

Die langwellige Emission über die Schnittstelle wird durch das XML-tag `InterfaceLongWaveEmission` beschrieben. Dieser Koeffizient beschreibt die langwellige Absorption und Emission über die Schnittstelle.



Beispiel 18. parameterdefinition fŸr langwellige Emission

```
<InterfaceLongWaveEmission modelType="Constant">
  <IBK:Parameter name="Emissivity" unit="---">0.9</IBK:Parameter>
</InterfaceLongWaveEmission>
```

Die `InterfaceLongWaveEmission` muss mit dem folgenden XML-Attribut `modelType` definiert werden.

Tabelle 10. Attribute

Attribut	Beschreibung	Format	Verwendung
<code>modelType</code>	Setzt den Typ des WŸrmeleitungsmodells  ¥ <code>Constant</code> - es wird ein konstantes Modell verwendet (derzeit die einzige Option)	key	erforderlich

Es kŸnnen XML-tags mit dem Namen `IBK:Parameter` mit den XML-Attributen `name` und `unit` mit den folgenden EintrŸgen definiert werden:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>Emissivity</code>	---	Konstanter Absorptionskoeffizient	0Œ1	erforderlich

9.4. Dampfdiffusion

#

MUSS SPŒTER DEFINIERT WERDEN.

Die Dampfdiffusion Ÿber die GrenzflŸche wird durch das XML-tag `InterfaceVaporDiffusion` beschrieben.

Beispiel 19. Parameterdefinition fŸr Dampfdiffusion

```
<InterfaceVaporDiffusion modelType="Constant">
  <IBK:Parameter name="VaporTransferCoefficient" unit="s/m">1</IBK:Parameter>
</InterfaceVaporDiffusion>
```

Das `InterfaceVaporDiffusion` muss mit dem folgenden XML-Attribut `modelType` definiert werden.

Tabelle 11. Parameter fŸr das `InterfaceVaporDiffusion`-tag



Attribut	Beschreibung	Format	Verwendung
<code>modelType</code>	Setzt den Typ des Wärmeleitungsmodells  ☒ <code>Constant</code> - es wird ein konstantes Modell verwendet (derzeit die einzige Option)	key	erforderlich

Es können XML-tags mit dem Namen `IBK:Parameter` mit den XML-Attributen `name` und `unit` mit den folgenden Einträgen definiert werden:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>VaporTransferCoefficient</code>	s/m	Dampfübergangskoeffizient	0.001-10.0	erforderlich

## 9.5. Luftstrom



MUSS SPÄTER DEFINIERT WERDEN.

Der Luftstrom über die Schnittstelle wird mit einem Druckkoeffizienten berechnet. Er wird im XML-tag `InterfaceAirFlow` beschrieben.

*Beispiel 20. Parameterdefinition für Luftstrom*

```
<InterfaceAirFlow modelType="Constant">
  <IBK:Parameter name="PressureCoefficient" unit="---">0.6</IBK:Parameter>
</InterfaceAirFlow>
```

Das `InterfaceAirFlow` muss mit dem folgenden XML-Attribut `modelType` definiert werden.

*Tabelle 12. Attribute*

Attribut	Beschreibung	Format	Verwendung
<code>modelType</code>	Setzt den Typ des Luftstroms  ☒ <code>Constant</code> - es wird ein konstantes Modell verwendet (derzeit die einzige Option)	key	erforderlich

Es können XML-tags mit dem Namen `IBK:Parameter` mit den XML-Attributen `name` und `unit` mit den folgenden Einträgen definiert werden:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
PressureCoefficient	---	Druckkoeffizient	0 ÷ 1	erforderlich

## 10. Aktive Schichten/Flächenheizungen

Eine Konstruktion kann thermisch wechselwirken mit andern Modelle, bspw. als Fußbodenheizung. Dafür muss im verwendeten Konstruktionstyp eine aktive Schicht definiert sein (siehe [Abschnitt 6.2.1](#)).

## 11. Eingebettete Objekte (Fenster, Türen, ...ffnungen)

Es kann mehrere Definitionen für eingebettete Objekte geben.

*Beispiel 21. Definition eines Fensters innerhalb einer Bauinstanz*

```
<ConstructionInstance id="1">
  <IBK:Parameter name="Area" unit="m2">12</IBK:Parameter>
  ...
  <EmbeddedObjects>
    <EmbeddedObject id="2000" displayName="Ein Fenster">
      <!-- Area-Parameter ist erforderlich. -->
      <IBK:Parameter name="Area" unit="m2">8</IBK:Parameter>
      ...
    </EmbeddedObject>
  </EmbeddedObjects>
</ConstructionInstance>
```

Eingebettete Objekte müssen mindestens einen Parameter **Area** definiert haben. Diese Fläche darf die Bruttofläche der Konstruktionsinstanz nicht überschreiten.

Ein eingebettetes Objekt wird durch eingebettete Datenobjekte weiter qualifiziert.

### 11.1. Fenster

Ein Fenster besteht aus einer Verglasung und optional einem Rahmen und Trennwänden. Ohne Rahmen und Trennwände sieht die Definition für ein solches Fenster wie folgt aus:

*Beispiel 22. Parameterdefinition für Basisfenster ohne Rahmen*

```
<EmbeddedObject id="2000" displayName="Ein Fenster">
  <IBK:Parameter name="Area" unit="m2">8</IBK:Parameter>
  <Window glazingSystemId="123"/>
</EmbeddedObject>
```

Nur das Verglasungssystem wird über die ID referenziert. Verglasungssysteme sind in der

Datenbankliste der Verglasungssysteme definiert, siehe [Abschnitt 6.3](#).

Das Fenster kann einen Rahmen und/oder Trennwände haben. Diese sind separate Entitäten, da das Material von Rahmen und Trennwänden (und damit die Wärmeleitfähigkeit zwischen diesen Materialien) unterschiedlich sein kann. Diese werden in den XML-tags **Frame** und **Divider** definiert:

*Beispiel 23. Parameterdefinition für Basisfenster mit Rahmen und Trennwand*

```
<EmbeddedObject id="2000" displayName="Ein Fenster">
  <IBK:Parameter name="Area" unit="m2">8</IBK:Parameter>
  <Window glazingSystemId="123">
    <Frame materialId="1001">
      <IBK:Parameter name="Area" unit="m2">3</IBK:Parameter>
    </Frame>
    <Divider materialId="1002">
      <IBK:Parameter name="Area" unit="m2">2</IBK:Parameter>
    </Divider>
  </Window>
</EmbeddedObject>
```

Die Materialeigenschaften (derzeit nur die Wärmeleitfähigkeit) von Rahmen- und Trennelementen werden aus dem über die ID referenzierten Material übernommen.

Die tatsächliche Geometrie von Rahmen- und Trennelementen ist nicht wichtig, aber ihre Gesamtquerschnittsfläche muss als Parameter **Area** angegeben werden.

■

Der von Rahmen und Trennwand belegte Querschnitt darf die Bruttofläche des eingebetteten Fensterobjekts nicht überschreiten. Die tatsächliche lichtdurchlässige Verglasungsfläche wird als Differenz zwischen der Fläche des eingebetteten Objekts und den Flächen von Rahmen und Trennwand berechnet.

!

Wenn die Größe des Fensters (oder des eingebetteten Objekts) geändert wird, müssen die Größen von Rahmen und Trennwand entsprechend angepasst werden. Es wäre zwar möglich gewesen, Rahmen- und Trennwandquerschnitte auch als relativen Prozentsatz zu definieren, dennoch muss dieser Prozentsatz bei einer Größenänderung des Fensters aktualisiert werden.

#### 11.1.1.1. Fensterverschattung

Es ist möglich, vorberechnete Verschattung bzw. Verschattungseinrichtungen sowohl auf opake als auch auf transluzente Fassadenelemente anzuwenden. Dabei wird zwischen geregelter/fester Verschattungsvorrichtung am Fenster und einer geometrischen Umgebungs-/Eigenverschattung unterschieden.

■

Die vorberechnete Umgebungsverschattung bzw. Eigenverschattung wird als globale Eigenschaft im **Location**-tag definiert (siehe [Abschnitt 12.4](#)).

Wenn eine vorberechnete Umgebungsverschattung definiert ist, wird für jede opake und transluzente Fläche ein Verschattungsgrad (Abminderungsfaktor) angegeben. Dieser wird automatisch bei der Strahlungsberechnung auf Flächen und Fenster einbezogen.

Die nachfolgend beschriebene (geregelte) Fensterverschattung wird zusätzlich berücksichtigt.



Wie im Abschnitt [Abschnitt 12.4](#) beschrieben, erfolgt die Zuordnung zwischen bereitgestellten Datenspalten und Objekt-ID über die eindeutige ID Nummer der Konstruktionsinstanz bzw. des eingebetteten Objekts.

Alternativ oder zusätzlich zur vorberechneten Umgebungsverschattung ist es möglich, eine geregelte Verschattung für das Fenster zu definieren.

Beispiel 24. Parameterdefinition für konstante Verschattung

```
<Window glazingSystemId="123">
  ...
  <Shading modelType="Constant">
    <IBK:Parameter name="ReductionFactor" unit="---">0.6</IBK:Parameter>
  </Shading>
</Window>
```

Das XML-tag **Shading** muss mit den folgenden XML-Attributen definiert werden:

Tabelle 13. Attribute

Attribut	Beschreibung	Format	Verwendung
modelType	Setzt den Typ des Schattierungsmodells  • <b>Constant</b> - Konstante Verschattung  • <b>Precomputed</b> - Zeitabhängige vorberechnete Verschattungsfaktoren  • <b>Controlled</b> - Verschattung wird in Abhängigkeit einer Strahlungsintensität	key	erforderlich
controlModelId	ID des Verschattungskontrollmodells	ID	erforderlich für Controlled

Tabelle 14. Child-Tags

Element	Beschreibung	Format	Verwendung
<code>PrecomputedReductionFactor</code>	Zeitreihe mit vorberechneten Abminderungsfaktoren infolge Verschattung (sollte für eine vorberechnete, geregelte Verschattung verwendet werden)		erforderlich für <code>Precomputed</code>

Es können XML-tags mit dem Namen `IBK:Parameters` mit den XML-Attributen `name` und `unit` mit den folgenden Einträgen definiert werden:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>ReductionFactor</code>	---	Prozentualer Anteil der verbleibenden solaren Gewinne, wenn die Beschattung geschlossen ist	0 ÷ 1	erforderlich für <code>Constant</code> und <code>Controlled</code>

Berechnung des Beschattungsfaktors auf Basis des Steuersignals

Nominale (maximale) `ReductionFactor` =  $z$  = 80%

$z$ -Wert in Abhängigkeit vom Steuersignal  $F_z$ :

$F_z = 1$  = voll beschattet:  $z = 1 - (1 - 80\%) * F_z = 0,8$   
 $F_z = 0$  = unverschattet verschattet:  $z = 1 - (1 - 80\%) * F_z = 1$   
 $F_z = 0,5$  = teilweise verschattet:  $z = 1 - (1 - 80\%) * F_z = 0,9$

## 12. Klimadaten und Standortinformationen

### 12.1. † bersicht

Klimalasten werden in NANDRAD über Klimadateien bereitgestellt. Für die Berechnung der Sonneneinstrahlung werden Informationen über den Gebäudestandort (in der Regel in der Klimadatei enthalten) sowie die Ausrichtung und Neigung der verschiedenen Konstruktionsflächen benötigt (definiert für Außenflächen, siehe [Abschnitt 8](#)).

### 12.2. Spezifikation

Informationen über Standort- und Klimadaten werden im Abschnitt `Location` der Projektdatei gespeichert:

### Beispiel 25. Definition des Standorts

```
<Location>
  <ClimateFilePath>${Projektverzeichnis}/climate/GER_Potsdam_2017.c6b</ClimateFilePath>
  <IBK:Parameter name="Latitude" unit="Deg">51</IBK:Parameter>
  <IBK:Parameter name="Longitude" unit="Deg">13</IBK:Parameter>
  <IBK:Parameter name="Albedo" unit="---">0.2</IBK:Parameter>
  <IBK:Parameter name="Altitude" unit="m">100</IBK:Parameter>
  <IBK:Flag name="PerezDiffuseRadiationModel">false</IBK:Flag>
</Location>
```

Die Außenklimabedingungen, einschließlich Standortinformationen werden aus einer Klimadatei bezogen, angegeben im Element `<ClimateFilePath>`. Dieser kann Platzhalter enthalten (siehe [Abschnitt 4](#)).

Zusätzliche Parameter (siehe [Abschnitt 3.1](#) für eine Beschreibung des Elementtyps `IBK:Parameter`):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Albedo	---	Wird für die Berechnung der diffusen Sonneneinstrahlung verwendet (siehe <a href="#">Abschnitt 12.3</a> )	0–1	erforderlich
(*)Altitude	m	wird für bestimmte höhenbezogene Parameter benötigt	>0	optional
Longitude	Deg	Wenn angegeben, wird der Ortsparameter Longitude der Klimadatendatei überschrieben (siehe <a href="#">Abschnitt 12.2.2</a> ).	-180–180	optional
Latitude	Deg	Wenn angegeben, wird der Ortsparameter Latitude der Klimadatendatei überschrieben (siehe <a href="#">Abschnitt 12.2.2</a> ).	-90–90	optional

(\*) wird noch nicht verwendet.

Flags und Optionen (siehe [Abschnitt 3.3](#) für eine Beschreibung des Elementtyps `IBK:Flag`):

Name	Beschreibung	Standard	Verwendung
PerezDiffuseRadiationModel	Legt fest, ob das Perez-Modell für die Berechnung der diffusen Sonnenstrahlung verwendet werden soll	false	optional
ContinuousShadingFactorData	Wenn gesetzt werden Verschattungsdaten als kontinuierliche Zeitreihen behandelt (siehe <a href="#">Abschnitt 12.4</a> )	false	optional

### 12.2.1. Klimadateien

Derzeit werden **c6b**, **wac** und **epw** Dateien unterstützt (siehe auch Dokumentation zur Software [CCM-Editor](#)).

Sie müssen den Pfad zur Klimadatei im `<ClimateFileName>`-Element angeben. Dabei können Sie einen absoluten oder relativen Pfad angeben.

Wenn ein relativer Pfad angegeben wird, wird dieser mit dem aktuellen Arbeitsverzeichnis als Referenz aufgelöst. Wenn Sie zum Beispiel angegeben haben

```
<ClimateFilePath>GER_Potsdam_2017.c6b</ClimateFilePath>
```

und der Solver aus dem Verzeichnis `/home/user/sim/Project1` gestartet wird, wird die Klimadatendatei in `/home/user/sim/Project1/GER_Potsdam_2017.c6b` gesucht. Wenn der Solver von einem anderen Verzeichnis aus gestartet wird, wird die referenzierte Klimadatendatei nicht gefunden und es wird eine Fehlermeldung ausgegeben.

Um dieses Problem zu vermeiden, können Sie Verzeichnissplatzhalter angeben, um den Pfad zur die Klimadatendatei *relativ* zum Speicherort der Projektdatei anzugeben. Der eingebaute Pfadplatzhalter `${Project Directory}` wird durch das Verzeichnis ersetzt, in dem sich die Projektdatei befindet. Verwenden Sie den Platzhalter einfach wie einen regulären Verzeichnisteil, z. B:

```
<ClimateFilePath>${Project Directory}/climate /GER_Potsdam_2017.c6b</ClimateFilePath>
```

Es ist möglich, für alle extern referenzierten Dateien eigene Platzhalter im Projekt zu definieren, siehe [Abschnitt 4](#).

### 12.2.2. Gebäude-/Klimastandort

Klimadatendateien enthalten Informationen über Breiten- und Längengrad der Wetterstation, die auch als Standort des Gebäudes angenommen wird. Dadurch wird sichergestellt, dass Simulationszeit und Sonnenstand übereinstimmen.

Es ist jedoch auch möglich, den Breitengrad/Längengrad in der Projektdatei anders zu definieren. Wenn diese Parameter in der Projektdatei angegeben werden (es müssen immer beide Parameter angegeben werden und gültig sein), werden diese Parameter aus der Projektdatei anstelle der Standortparameter der Klimadatei verwendet.



Durch die Angabe eines von der Klimastation abweichenden Breitengrades kann der berechnete Sonnenstand nicht mehr mit dem Sonnenstand an der Wetterstation übereinstimmen, was zu möglicherweise falschen Solarstrahlungslasten führt.

Gültiger Wertebereich für **Latitude** ist `[-90,90]` Grad (positive Werte entsprechen der nördlichen Hemisphäre), für **Longitude** ist es `[-180,180]` Grad (positive Werte sind östlich von Greenwich).

### 12.2.3. Zyklische (j hrliche) und kontinuierliche (mehrj hrige) Klimadaten

Die Klimadaten-Datei kann 8760 Stundenwerte f r ein ganzes Jahr enthalten. Anderfalls werden die Klimadaten als kontinuierlich abgelegte Daten f r Zeitwerte in einem beliebigen Zeitbereich betrachtet. Dabei k nnen die Klimadaten auch mit variierenden Zeitabst nden zwischen den Datenpunkten definiert sein. Solche Klimadateien k nnen nicht f r die j hrliche/zyklische Berechnung verwendet werden, sondern ben tigen ein bestimmtes (passendes) Simulationszeitintervall (siehe [Abschnitt 16.1.1](#)).

#### Zyklisches Jahresklima

Hierbei werden die Klimadaten in Stundenwerten bereitgestellt. Die Interpretation dieser Werte h ngt von der Art der physikalischen Gr  e ab. NANDRAD unterscheidet zwischen *Zustandsgr  en* und *Fluss-/Lastgr  en*.

Zustandsgr  en sind:

-   Temperaturen
-   relative Luftfeuchten
-   Luftdr cke
-   Windrichtung
-   Windgeschwindigkeit

Fluss-/Lastgr  en sind:

-   direkte Sonnenstrahlungsintensit t (in Normalrichtung der Sonne)
-   diffuse solare Strahlungsintensit t (in horizontaler Ebene)
-   Regenlast
-   langwellige Himmelsemission/-gegenstrahlung

Es wird erwartet, dass die Zustandsgr  en als *Momentanwerte* am *Ende jeder Stunde* angegeben werden. Subst ndliche Werte werden durch lineare Interpolation erhalten, wie in [Abbildung 2](#) gezeigt.



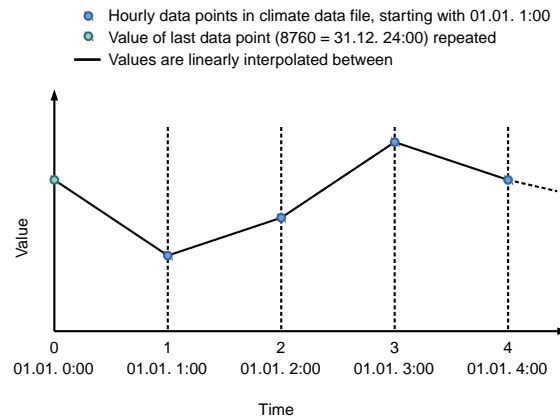


Abbildung 2. Bei Zustandsgrößen wird eine lineare Interpolation verwendet, um den Zeitverlauf zwischen den stündlichen Momentanwerten zu rekonstruieren

Fluss-/Lastgrößen werden als Mittelwerte über die letzte Stunde erwartet. Die substündlichen Werte werden durch lineare Interpolation zwischen den in der Mitte jeder Stunde platzierten Mittelwerten erhalten, wie in [Abbildung 3](#) gezeigt.

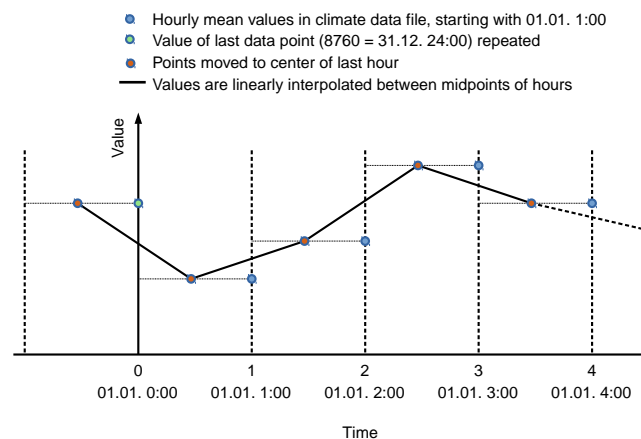


Abbildung 3. Bei Flussgrößen/Lasten werden die Werte in die Stundenmitte verschoben und dann linear interpoliert.

!

Die so erhaltenen Integralwerte in einer Stunde weichen leicht von den ursprünglichen integralen Mittelwerten ab (siehe z. B. Stunde zwischen 2:00 und 3:00). Die Fehler sind jedoch klein und heben sich im Tagesverlauf fast komplett auf. Dafür sind die generierten Zeitreihen und substündlichen Werte stets stetig.

## Kontinuierliche Daten

Hierbei enthält die Klimadatendatei Datenpunkte (mindestens 2), wodurch auch der früheste Start- und späteste Endpunkt der Simulation definiert wird.



Wenn Sie die Simulation über die verfügbaren Klimadaten hinaus fortsetzen, werden die letzten Werte im Klimadatensatz konstant gehalten. Dies führt in der Regel zu sinnlosen Ergebnissen (es sei denn, dies ist in künstlichen Testfällen beabsichtigt).

Da der Benutzer in den Klimadatendateien beliebige Zeitschritte bis hin zu winzigen Werten wählen kann, hängt die Genauigkeit der Eingabedaten von den Benutzereingaben ab. Zwischen den Zeitpunkten wird der Solver alle Größen in der Klimadatendatei linear interpolieren und nicht wie bei stündlichen Daten zwischen Zuständen und Lasten unterscheiden.



Um das gleiche Ergebnis wie bei jährlichen, zyklischen Stundendaten zu erzielen, müssen Klimadaten in 30-Minuten-Intervallen angegeben und interpolierte Werte am Ende und in der Mitte jeder Stunde selbst berechnet werden.

#### 12.2.4. Zusätzliche Strahlungssensoren

Es ist möglich zusätzliche Ebenen (Sensoren) zu spezifizieren, um Strahlungslasten zu berechnen und für andere Modelle als Sensorgrößen zur Verfügung zu stellen. Dies geschieht durch die Angabe einer **Sensor**-Definition.

##### Beispiel 26. Definition von Sensoren

```
<Location>
  ...
  <Sensors>
    <!-- Flachdach>
    <Sensor id="1">
      <IBK:Parameter name="Orientation" unit="Deg">0</IBK:Parameter>
      <IBK:Parameter name="Inclination" unit="Deg">0</IBK:Parameter>
    </Sensor>
    <!-- Nordwand 90 -->
    <Sensor id="2">
      <IBK:Parameter name="Orientation" unit="Deg">0</IBK:Parameter>
      <IBK:Parameter name="Inclination" unit="Deg">90</IBK:Parameter>
    </Sensor>
    ...
  </Sensors>
</Location>
```

Tabelle 15. Attribute

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Kennung des Sensors	int	erforderlich

Parameter (siehe Abschnitt [Abschnitt 3.1](#) für eine Beschreibung des Elementtyps **IBK:Parameter**):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<b>O</b> ri <b>e</b> ntat <i>ion</i>	Deg	Ausrichtung des Sensors	0°-360	erforderlich
<b>I</b> nc <i>l</i> inat <i>ion</i>	Deg	Neigung des Sensors  0° - nach oben gerichtet 90° - z. B. wie eine senkrechte Wand 180° - nach unten gerichtet	0°-180	erforderlich

Einem Sensor muss eine eindeutige ID-Nummer und die obligatorischen Parameter **O**ri **e**ntat*ion* und **I**nc*l*inat*ion* gegeben werden (siehe [Abschnitt 8](#) für Details zu deren Definition).

Für jeden Sensor werden 4 Ausgangsgrößen erzeugt:

- 0° **D**i **r**ect*S*W*R*ad*O*n*P*l*a*ne[<sensor id>] - direkte Sonnenstrahlungsintensität auf die Sensorfläche in [W/m<sup>2</sup>]
- 0° **D**i **f**fuse*S*W*R*ad*O*n*P*l*a*ne[<sensor id>] - diffuse Sonnenstrahlungsintensität auf die Sensorfläche in [W/m<sup>2</sup>]
- 0° **G**l*o*bal *S*W*R*ad*O*n*P*l*a*ne[<sensor id>] - globale Strahlungsintensität auf die Sensorfläche in [W/m<sup>2</sup>] (die Summe der beiden erstgenannten Größen)
- 0° **I**nci **d**ence*A*ngl*e*On*P*l*a*ne[<sensor id>] - der Sonneneinfallswinkel auf die Sensorfläche in [Grad] (0°, wenn der Sonnenstrahl senkrecht zur Ebene steht, 90°, wenn der Strahl parallel zur Ebene verläuft oder wenn die Sonne unter dem Horizont ist)

*Beispiel 27. Ausgabedefinition für Sensorwerte (siehe auch Beschreibung von Ergebnisdefinitionen in [Abschnitt 15](#)).*

```

<OutputDefinitionen>
  ...
  <!-- direkte Strahlung intensiv vom Sensor mit id=2 -->
  <OutputDefinition>
    <Quantity>DirektSWRadOnPlane[2]</Quantity>
    <ObjectName>Location</ObjectName>
    <GridName>minYtlIch</GridName>
  </OutputDefinition>
  <!-- Einfallswinkel vom Sensor mit id=42 -->
  <OutputDefinition>
    <Quantity>IncidenceAngleOnPlane[42]</Quantity>
    <ObjectName>Location</ObjectName>
    <GridName>minYtlIch</GridName>
  </OutputDefinition>
  ...
</OutputDefinitions>

```

## 12.3. Sonnenstrahlungsberechnung

Die Berechnung der Sonneneinstrahlung folgt den in der *Physikalischen Modellreferenz* aufgeführten Gleichungen. Der Parameter **Albedo** wird bei der Berechnung der diffusen Strahlungslast verwendet. Die Schalter **PerezDiffuseRadiationModel** beeinflusst die Berechnung der Diffusstrahlung.

## 12.4. Vorberechnete externe Verschattung/Eigenverschattung

In einem vorgelagerten Rechenschritt kann für jedes Flächenelement des Gebäudes der Anteil der sonnenbeschienenen Fläche berechnet werden. In [Abbildung 4](#) wird zum Beispiel eine Fassade teilweise verschattet.

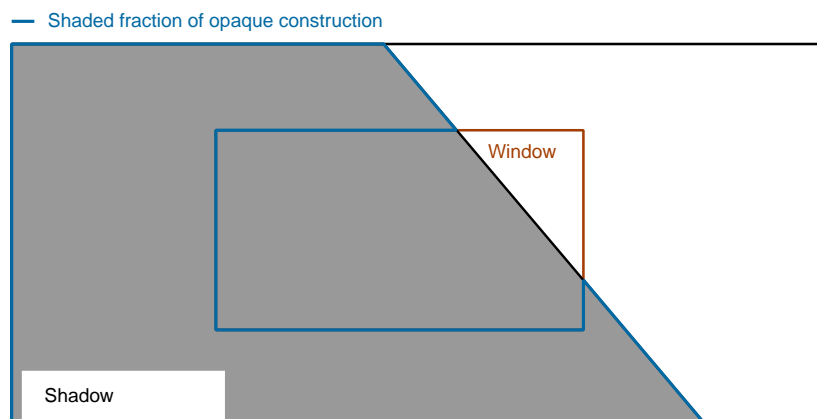


Abbildung 4. Darstellung einer teilverschatteten Fassade mit einem Fenster

Die Software kann nun den Prozentsatz der verschatteten Fläche sowohl für das opake Fassadenelement als auch für das Fensterobjekt separat berechnen. Das Fenster ist zu ca. 80 % verschattet, und ca. 20 % der opaken Fläche sind noch der Sonne ausgesetzt. Der letztere Anteil wird auch als *Sonnenlichtfaktor* bzw. *Abminderungsfaktor infolge Verschattung* (engl. *shading factors*) bezeichnet.

Der für eine opaque Konstruktion gespeicherte Faktor ist immer *exklusive* aller eingebetteter Objekte zu verstehen. [Abbildung 5](#) zeigt ein ähnliches Beispiel, bei dem die Berechnung der Flächen und Sonnenlichtfaktoren erläutert wird.

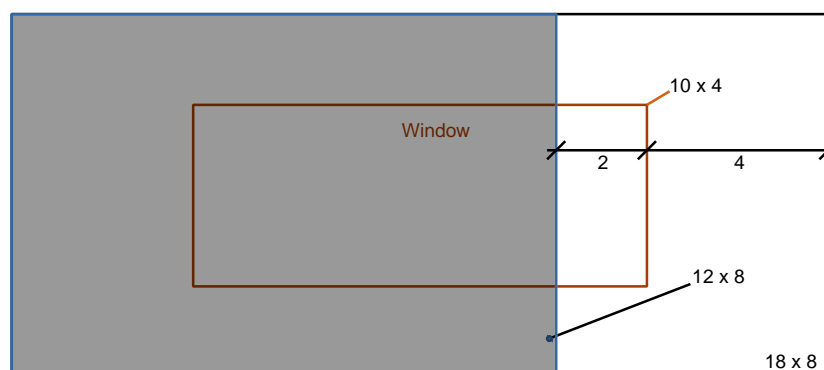


Abbildung 5. Berechnungsbeispiel für eine teilverschattete Fassade mit einem Fenster

Die Konstruktion hat eine Fläche von  $18 \times 8 = 144 \text{ m}^2$ . Das Fenster hat eine Fläche von  $10 \times 4 = 40 \text{ m}^2$ . Damit verbleibt für die eine opake Konstruktionsfläche von  $144 - 40 = 104 \text{ m}^2$ .

Der Schatten auf dem Fenster allein nimmt  $8 \times 4 = 32 \text{ m}^2$  ein. Der Sonnenlichtfaktor für das Fenster allein beträgt also  $1 - 32/40 = 20\%$ .

Die verschattete Fläche auf der opaken Konstruktion beträgt  $12 \times 8 - 8 \times 4 = 96 - 32 = 64 \text{ m}^2$ . Der Sonnenlichtfaktor, der in der Gleichung für die Belastung durch die Sonneneinstrahlung verwendet werden muss beträgt also  $1 - 64/104 = 38,5 \%$ .

Die Werte 0.385 und 0.2 werden in der Datei für die vorberechneten Sonnenlichtfaktoren gespeichert.

Die mittlere Strahlungsintensität auf eine opake Fläche ergibt sich dann aus:

Mittlere direkte Strahlungslast in  $[\text{W}/\text{m}^2] = \text{Sonnenlichtfaktor (aus Datei)} * \text{direkte Strahlungslast}$

#### 12.4.1. Dateiformat für vorberechnete Sonnenlichtfaktoren

Die Datei mit den vorberechneten Sonnenlichtfaktoren wird im XML-Element `Location` definiert, im Kind-Element `ShadingFactorFilePath`. Der hier angegebene Pfad kann ein absoluter Pfad oder ein relativer Pfad sein, der einem Platzhalter folgt (bspw. `${Project Directory}`), siehe [Abschnitt 4](#).

Die Datei kann als `tsv` Datei oder DataIO-Datei (ASCII oder Binärformat, Endungen `d60` und `d6b`) bereitgestellt werden.

Die Dateien enthalten für bestimmte, kontinuierlich ansteigende Zeitpunkte die jeweils berechneten Sonnenlichtfaktoren.



Bei der Berechnung werden die Werte in der Datentabelle linear interpoliert.

Standardmäßig wird von zyklischen Jahresdaten ausgegangen. Dabei müssen die Zeitpunkte stets  $< 365 \text{ d}$  bleiben. Sollen kontinuierliche Daten verwendet werden, muss der Schalter `ContinuousShadingFactorData` eingeschaltet sein.

#### TSV-Format für Sonnenlichtfaktor-Dateien

Bei Verwendung des `tsv`-Formats müssen die Regeln des `tsv`-Dateiformats (siehe *PostProc 2* Dokumentation) eingehalten werden. Es gibt eine einzelne Kopfzeile. Die erste Spalte ist die Zeitspalte mit Zeit-offsets relativ zu Mitternacht des 1. Januar des Startjahres. Es können beliebige Zeiteinheiten verwendet werden.

Alle anderen Spalten enthalten die berechneten Sonnenlichtfaktoren, wobei jeder Spaltenkopf die jeweilige Fläche mit eindeutiger ID identifiziert. Für opake Flächen werden die IDs der jeweiligen Konstruktionsinstanzen verwendet. Bei Fenstern werden die IDs der eingebetteten Objekte verwendet. Als Werteeinheit muss `---` verwendet werden.

### Beispiel 28. TSV-Datei mit Sonnenlichtfaktoren für 4 Flöchen

```
Time [d] 1001 [---] 1002 [---] 1003 [---] 1004 [---]
0 1 1 1 1
181 1 1 1 1
182 0 0 1 1
185 0 0 1 1
186 0 1 1 1
188 0 1 1 1
189 1 1 1 1
```

Fenster/Konstruktion mit IDs 1001 und 1002 werden in den Tagen 182 bis 188 verschattet. Die Fenster 1003 und 1004 bleiben die ganze Zeit unverschattet.

### DataIO Format

Bei Verwendung des DataIO-Formats muss das REFERENCE-Format verwendet werden. Das Feld INDICES enthält die IDs der jeweiligen Flöchen.

### Beispiel 29. DataIO-Datei mit Sonnenlichtfaktoren für 4 Flöchen, analog zum obigen TSV-Beispiel

```
D60ARLZ! 007.000
TYPE      = REFERENCE
QUANTITY  = 1001 | 1002 | 1003 | 1004
VALUE_UNIT = ---
TIME_UNIT = d
INDICES    = 1001 1002 1003 1004

0 1 1 1 1
181 1 1 1 1
182 0 0 1 1
185 0 0 1 1
186 0 1 1 1
188 0 1 1 1
189 1 1 1 1
```

Für größere Gebäude ist das binäre DataIO-Format (mit gleichem Inhalt) zu empfehlen.

## 13. Objektlisten und Ergebnisreferenzen

Wann immer es notwendig ist, ein Berechnungsergebnis (eines Modellobjekts) zu referenzieren, geschieht dies über die *ObjectLists*.

In NANDRAD werden physikalische Gleichungen in Form von Modellobjekten organisiert, zum Beispiel Zonen oder Konstruktionen. Diese Modellobjekte können durch einen Modelltyp und eine ID-Nummer eindeutig identifiziert werden. Zum Beispiel werden alle für einen Raum/eine Zone berechneten Größen durch den Modelltyp *Zone* und die ID-Nummer der jeweiligen Zone identifiziert. [Tabelle 16](#) listet die verfügbaren Referenztyp-Schlüsselwörter auf.

Tabelle 16. Modell-Referenztypen

Schlüsselwort	Beschreibung
Zone	Variablen bezogen auf den Raum (thermische Zonen)
ConstructionInstance	Variablen, die sich auf Konstruktionen beziehen
Schedule	Geplante Parameter
Location	Variablen aus dem Klimaberechnungsmodell, einschließlich Strahlungssensorwerte
Model	Modellspezifische Variablen/Ergebnisse

Beispiel 30 zeigt mehrere Beispiele für Definitionen der Objektliste.

Beispiel 30. Definition von mehreren Objektlisten

```

<ObjectLists>
  <ObjectList name="All_zones">
    <FilterID>*</FilterID>
    <ReferenceType>Zone</ReferenceType>
  </ObjectList>
  <ObjectList name="Zone_Var01">
    <FilterID>1</FilterID>
    <ReferenceType>Zone</ReferenceType>
  </ObjectList>
  <ObjectList name="Wall_1_and_2">
    <FilterID>1,2</FilterID>
    <ReferenceType>ConstructionInstance</ReferenceType>
  </ObjectList>
  <ObjectList name="InfiltrationModel">
    <FilterID>501</FilterID>
    <ReferenceType>Model</ReferenceType>
  </ObjectList>
  ...
</ObjectLists>

```

## 13.1. Objektlisten-Definitionen

Alle Objektlisten werden innerhalb des übergeordneten tags **ObjectLists** definiert. Jede Objektlisten-Definition beginnt mit dem XML-tag **ObjectList** mit dem obligatorischen Attribut **name**, das die Objektliste eindeutig identifiziert.

Das XML-tag **ObjectList** hat die folgenden untergeordneten tags.

Tabelle 17. Modell-Referenztypen

Schlüsselwort	Beschreibung
FilterID	ID-Filtermuster (siehe Beschreibung unten)
ReferenceType	Modellobjekt-Referenztyp (siehe Tabelle 16)

## 13.2. ID-Filter-Muster

Objekte (mit gleichem **ReferenceType**) werden durch ihre ID-Nummer eindeutig identifiziert.



ID-Nummern müssen nur für Objekte mit gleichem **ReferenceType** eindeutig sein. Daher ist es möglich, Zone #1 und ConstructionInstance #1 gleichzeitig zu definieren.

Ein Filtermuster kann aus mehreren Teilen bestehen, die durch , (Komma) getrennt sind, zum Beispiel: **1, 4, 13-20**. Jeder Teil kann das folgende Format haben:

¥ eine einzelne ID-Nummer, z. B. *12*

¥ ein Bereich von ID-Nummern, z. B. *1-100*

¥ \* (wählt alle IDs aus)

Wenn IDs mehrmals angegeben werden, z.B. in "3, 1-10", enthält die resultierende ID-Menge jede ID nur einmal.

## 14. Zeitpläne

### 14.1. Übersicht

Zeitpläne liefern rein zeitabhängige Größen, ähnlich wie Klimabelastungen. Im Unterschied zu anderen ergebnisproduzierenden Modellen erzeugen Zeitpläne Variablen für Mengen von abhängigen Modellen. Als solches wird ein Zeitplan für eine Objektliste formuliert, die eine Menge von Objekten mit den vom Zeitplan vorgegebenen Werten auswählt. Sie werden z. B. in den folgenden Objekten oder Objektlisten verwendet:

¥ Belegungsraten, Wärmelasten, Bekleidungsfaktoren im Personenlastmodell.

¥ Heiz-/Kühlsolltemperaturen für Thermostatsteuerungen

¥ Massendurchflussraten oder Temperatursollwerte für Anlagenteile

¥ Elektrische Leistungsraten für Beleuchtung und elektrische Geräte

Ein Zeitplan definiert z. B. einen Heizungssollwert **HeatingSetPoint** für bestimmte Zonen wie z. B. Wohnräume. Diese werden über eine Objektliste mit dem Namen "Living room" ausgewählt, die Objekte vom Typ **Zone** und einem bestimmten ID-Bereich (wird später noch genauer beschrieben) auswählt.

Es gibt zwei Möglichkeiten, einen Zeitplan zu beschreiben:

¥ **ScheduleGroups**

¥ **Annual Schedules**.

Die beiden Möglichkeiten werden in **Tagesschema-basierte Zeitpläne** und **Jahresschaltpläne** detailliert besprochen.

Außerdem können Zeitplandaten auf zwei verschiedene Arten behandelt werden, als



¥ Zyklische Daten und

¥ Nicht-zyklische Daten.

Zyklische Daten bedeutet, dass die Zeitplanwerte nach dem Ende der Zeitplanperiode wiederholt werden. Das bedeutet zum Beispiel, dass ein jährlicher Zeitplan zweimal ausgeführt wird, wenn die Simulationszeit auf zwei Jahre eingestellt wird. Zyklische Daten können durch **ScheduleGroups** und **Annual Schedules** definiert werden.

Nicht-zyklische Daten werden immer nur einmal verwendet. Dies ist sinnvoll, wenn gemessene Daten (Monitoring) zum Einrichten von Zeitplänen verwendet werden sollen. Dann muss die Simulation nur für die Zeitspanne eingestellt werden, in der die gemessenen Daten vorhanden sind. Nicht-zyklische Daten können nur durch **Annual Schedules** definiert werden.

### Beispiel 31. Definition eines Zeitplans

```
<Schedules>
  <HolidayDays>5,10</HolidayDays>
  <WeekEndDays>Fri,Sat</WeekEndDays>
  <FirstDayOfYear>Fri</FirstDayOfYear>
  <IBK:Flag name="EnableCyclicSchedules">true</IBK:Flag>

  <ScheduleGroups>
    ...
  </ScheduleGroups>
  <Annual Schedules>
    ...
  </Annual Schedules>
</Schedules>
```

Innerhalb des Objekts **Schedules** können auch die folgenden XML-tags angegeben werden

¥ **FirstDayOfYear**

¥ **HolidayDays**

¥ **WeekEndDays**

¥ **Schedule**

¥ **IBK:Flag** mit dem Namen **EnableCyclicSchedules**

Tabelle 18. mögliche Definitionen von XML-tags

XML-tags	Beschreibung	Format	Verwendung
<code>FirstDayOfYear</code>	<p>Der Tagestyp des 1. Januar (Offset des Wochentages des Startjahres.</p> <p> <code>Mon</code> - Monday (<i>Standard</i>)  <code>Tue</code> - Tuesday  <code>Wed</code> - Wednesday  <code>Thu</code> - Thursday  <code>Fri</code> - Friday  <code>Sat</code> - Saturday  <code>Sun</code> - Sunday </p>	string	<i>optional</i>
<code>Holidays</code>	Liste der Feiertage, gespeichert in einer kommagetrennten Liste von Zahlen. Jede Zahl stellt den "Tag des Jahres" dar. Schalttage werden nicht eingeschlossen.	string	<i>optional</i>
<code>WeekEndDays</code>	Wochenendtage.	string	<i>optional</i>
<code>IBK: Flag</code>	<p><code>Name EnableCyclicSchedules</code></p> <p># Wenn auf true gesetzt, werden die Zeitpläne nach einem Jahr wiederholt.</p> <p># Wenn auf false gesetzt (gilt nur für jährliche Zeitpläne), werden diese jährlichen Zeitpläne nur einmal abgetastet.</p>	<i>true (Standard) / false</i>	<i>optional</i>

## 14.2. Zeitplangruppen

`ScheduleGroup`-tags gibt es sowohl in täglichen schema-basierten Zeitplänen als auch in jährlichen Zeitplänen. Sie identifizieren die Zielobjekte für geplante Parameter. Um Mehrdeutigkeiten zu vermeiden, ist es nicht erlaubt, mehrere Zeitplangruppen mit derselben Objektliste zu definieren.



Definieren Sie nicht mehrere `ScheduleGroup`-Elemente mit der gleichen Objektliste!

## 14.3. Tagesschema-basierte Zeitpläne

Regelmäßige Zeitpläne werden auf Basis eines Tagesschemas definiert. Einige Parameter müssen wie in dem nachfolgenden XML-tag definiert werden.

*Beispiel 32. Definition einer Zeitplangruppe*

```
<ScheduleGroups>
  <ScheduleGroup objectList="All Zones">
```

```

É      <!-- AllDays constant -->
É      <Schedule type="AllDays">
É          ...
É      </Schedule>
É      <Schedule type="WeekDays">
É          ...
É      </Schedule>
É  </ScheduleGroup>
</ScheduleGroups>

```

Beispiel 33. ObjectList-Definition, die Zonenobjekte auswählt und "All Zones" heißt

```

<ObjectLists>
É  <ObjectList name="All Zones">
É    <FilterID>*</FilterID>
É    <ReferenzTyp>Zone</ReferenzTyp>
É  </ObjectList>
</ObjectLists>

```

Regelmäßige Zeitpläne werden innerhalb des XML-tags **ScheduleGroup** mit einem obligatorischen XML-Attribut namens **objectList** definiert, das namentlich auf eine **ObjectList** verweist (siehe [Tabelle 19](#)):

Tabelle 19. Attribut für die ScheduleGroup

Name	Beschreibung	Format	Verwendung
<b>objectList</b>	Verweise auf eine Objektliste mit dem angegebenen Namen	string	erforderlich

[Beispiel 32](#) zeigt eine solche Definition und [Beispiel 33](#) die entsprechende Objektliste.

### 14.3.1. Tägliche Zyklen

Innerhalb der **ScheduleGroup** können mehrere Objekte namens **Schedule** definiert werden. Die **Schedule**-Objekte benötigen ein XML-Attribut namens **type** mit unterschiedlichen Namen für bestimmte Tagestypen (siehe [Tabelle 20](#)). Innerhalb einer **ScheduleGroup** dürfen sich nicht zwei **Schedule**-Objekte mit demselben **type** befinden. Innerhalb jedes **Schedule**-Objekts wird ein Zeitplan definiert, der für alle Tage des angegebenen **Type** im Laufe eines ganzen Jahres gilt. Bei der Konstruktion von Zeitplänen gelten die folgenden Regeln.

In der ersten Priorität werden beim Typ **AllDays** angegebene tägliche Zeitplanwerte (z.B. **HeatingSetPoint**) auf alle Tage des ganzen Jahres gesetzt (Priorität 0). [Beispiel 34](#) zeigt eine solche Zeitplandefinition.

Danach überschreiben die **Types** **WeekEnd** und **WeekDay**, falls definiert, die bereits definierten Zeitplanwerte nur für alle Wochentage oder Wochenendtage (Priorität 1). Weiterhin definieren die Wochentage namens **Monday**, **Tuesday**, etc. für welche Tage die Zeitplanwerte wieder überschrieben werden (Priorität 2). Weiter geht es mit dem Tagestyp **Holiday** (Priorität 3) für die angegebenen Feiertage

innerhalb des Objekts **Hol i days**.

Es ist möglich, unterschiedliche Zeitpläne für einzelne Zeiträume des Jahres zu definieren, z. B. für das reguläre Jahr und die Sommerferien etc. Auf diese Weise kann ein Zeitplan für das gesamte Jahr definiert werden.

Beispiel 34. Zeitplandefinition mit Typ "AllDays"

```
<ScheduleGroup objectList="Zone01">
  <!-- Konstante "AllDays" -->
  <Schedule type="AllDays">
    <DailyCycles>
      <DailyCycle interpolation="Constant">
        <ZeitPunkte>0</ZeitPunkte>
        <Werte>InfiltrationRateSchedule [1/h]:0</Werte>
      </DailyCycle>
    </DailyCycles>
  </Schedule>
</ScheduleGroup>
```

Tabelle 20 zeigt die Tagestypen und die dazugehörigen Prioritäten.

Tabelle 20. Beschreibung des Attributs "Schedule Type"

Type	Priority	Description
AllDays	0	Werte werden auf alle Tage der Periode gesetzt
WeekEnd	1	Werte werden auf alle Wochenendtage des Zeitraums gesetzt
WeekDay	1	Werte werden auf alle Wochentage des Zeitraums gesetzt
Monday	2	Werte werden auf alle Montage des Zeitraums gesetzt
Tuesday	2	Werte werden auf alle Dienstag des Zeitraums gesetzt
Wednesday	2	Werte werden auf alle Mittwoch des Zeitraums gesetzt
Thursday	2	Werte werden auf alle Donnerstag des Zeitraums gesetzt
Friday	2	Werte werden auf alle Freitag des Zeitraums gesetzt
Saturday	2	Werte werden auf alle Samstag des Zeitraums gesetzt
Sunday	2	Werte werden auf alle Sonntag des Zeitraums gesetzt
Holiday	3	Werte werden auf alle Feiertage des Zeitraums gesetzt, die im tag <b>hol i days</b> angegeben sind

Beispiel 35 veranschaulicht die Verwendung verschiedener Zeitpläne zur Definition eines Wochenplans. Zunächst wird der grundlegende tägliche Zeitplan definiert. Dann werden spezielle Regeln für Dienstag und Wochenenden definiert. Abbildung 6 veranschaulicht den resultierenden Zeitplan.

### Beispiel 35. Zeitplandefinition mit verschiedenen Tagestypen

```
<Schedules>
  <WeekEndDays>Sat, Sun</WeekEndDays>
  <ScheduleGroups>
    <ScheduleGroup objectList="All zones">
      <!-- jeden Tag zwischen 8-10 -->
      <Schedule type="All Days">
        <DailyCycles>
          <DailyCycle interpolation="Constant">
            <TimePoints>0 6 10</TimePoints>
            <Values>InfiltrationRateSchedule [1/h]:0 0.4 0</Values>
          </DailyCycle>
        </DailyCycles>
      </Schedule>
      <!-- Dienstag keine Lüftung -->
      <Schedule type="Tuesday">
        <DailyCycles>
          <DailyCycle interpolation="Constant">
            <TimePoints>0</TimePoints>
            <Values>InfiltrationRateSchedule [1/h]:0</Values>
          </DailyCycle>
        </DailyCycles>
      </Schedule>
      <!-- Wochenende nur am Nachmittag -->
      <Schedule type="WeekEnd">
        <DailyCycles>
          <DailyCycle interpolation="Constant">
            <TimePoints>0 14 16</TimePoints>
            <Values>InfiltrationRateSchedule [1/h]:0 0.1 0</Values>
          </DailyCycle>
        </DailyCycles>
      </Schedule>
    </ScheduleGroup>
  </ScheduleGroups>
</Schedules>
```

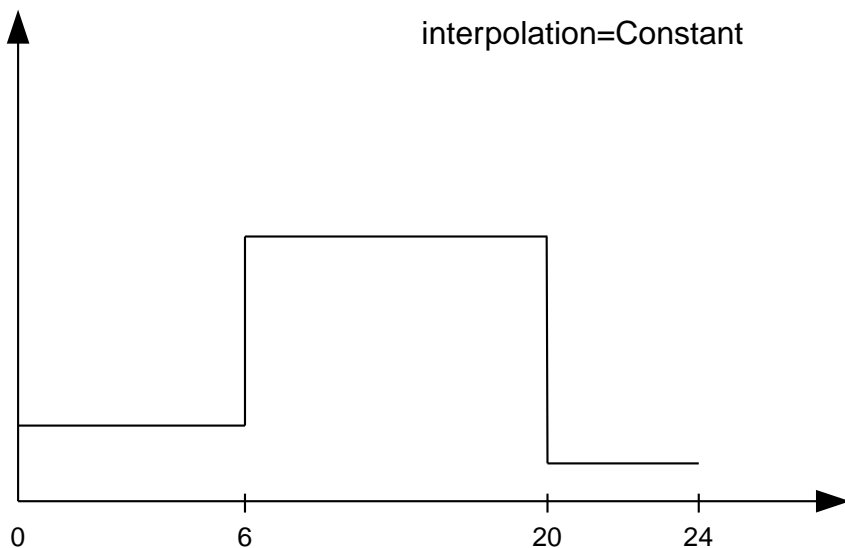


Abbildung 6. Abbildung des wöchentlichen Zeitplans, definiert durch [Beispiel 35](#)

### 14.3.2. DailyCycle Zeitintervalle

Ein **DailyCycle** definiert, wie sich eine oder mehrere Größen im Laufe des Tages ändern. Das untergeordnete tag **TimePoints** definiert durch Leerzeichen getrennte Zeitpunkte in Stunden [h] und damit die verschiedenen Zeitintervalle des Tages.

Wenn das Attribut **interpolation Constant** ist, dann gelten die folgenden Regeln:

- ¥ die Zeitpunkte werden als Startzeit des nächsten Intervalls interpretiert
- ¥ der erste Zeitpunkt muss immer 0 sein, der letzte muss < 24 h sein,
- ¥ der entsprechende Wert wird während dieses Intervalls als konstant angenommen

Ein Zeitpunktvektor "0 6 20" definiert z. B. drei Intervalle: 0-6, 6-20, 20-24 und die Datentabelle muss genau 3 Werte enthalten.

Wenn das Attribut **interpolation Linear** ist, dann gelten die folgenden Regeln:

- ¥ die Zeitpunkte sind Punkte in der Zeit, an denen zugehörige Werte gegeben sind
- ¥ der erste Zeitpunkt muss immer 0 sein, der letzte muss < 24 h sein, denn im zyklischen Betrieb ist der Zeitpunkt bei 24 h derselbe wie bei 0 h (und damit auch die geplanten Werte)
- ¥ zwischen den Zeitpunkten werden die Werte linear interpoliert

[Abbildung 7](#) und [Abbildung 7](#) zeigen den resultierenden Werteverlauf für die Zeitintervalle 0, 6, 20 und die entsprechenden Parameterwerte 2, 7, 1.

*Abbildung 7. Tageszyklus mit konstantem Interpolationsmodus*

Abbildung 8. Tageszyklus mit linearem Interpolationsmodus

!

Bei der Verwendung des linearen Interpolationsmodus wird der Wert um 24 Uhr vom Beginn des nächsten Tageszyklus genommen, der im Zeitplan definiert ist. Zum Beispiel würde in [Abbildung 6](#) der Wert am Montag 24:00 Uhr aus dem Zeitplan für Dienstag genommen werden, während der Wert am Mittwoch 24:00 Uhr aus dem regulären Zeitplan *AllDays* genommen würde.

!

Um ein einzelnes Intervall für den ganzen Tag zu definieren, geben Sie einfach "0" als Wert im XML-tag *TimePoints* an.

### 14.3.3. Tägliche Zyklusparameterwerte

Für jedes im tag *TimePoints* angegebene Intervall können eine oder mehrere Größen mit zugehörigen Einheiten angegeben werden. Dies geschieht durch die Definition der Datentabelle im XML-Tochtertag *Values* des *DailyCycle*-tags. Die Daten der Datentabelle werden wie folgt formatiert:

```
quantity1 [unit]:val 11 val 12 val 13; quantity2 [unit]:val 21 val 22 val 23; ...
```

Grundsätzlich wird jede physikalische Größe in einem string kodiert, wobei die strings für verschiedene Größen zu einem string mit ; (Semikolon) als Trennzeichen zusammengefasst werden.

Jeder Mengenstring setzt sich aus einem Header und den eigentlichen Werten zusammen. Die Werte sind einfach durch Leerzeichen/Tabs oder Komma getrennte Werte (Dezimalzahlen werden mit . (Punkt) als Dezimaltrennzeichen geschrieben).

Der Header ist ein Mengestichwort (siehe auch [Variablenliste](#)), gefolgt von seiner Einheit in Klammern. So hat z. B. eine Heizungssolltemperatur die Kopfzeile *HeatingSetPointTemperature [C]* und die Werte werden dann in Grad C angegeben.

Es müssen *exakt* so viele Werte angegeben werden, wie es Zeitpunkte im XML-tag *TimePoints* gibt. In dieser Datentabelle können Sie so viele Größen angeben, wie Sie benötigen.

Beispiel 36 zeigt einen Tageszyklus mit zwei geplanten Mengen und drei Intervallen.

#### Beispiel 36. Tageszyklus mit zwei disponierten Mengen

```
<DailyCycle interpolation="Constant">
  <TimePoints>0 6 10</TimePoints>
  <Values>
    InfiltrationRateSchedule [1/h]: 0 0.4 0;
    HeatingSetPointTemperature [C]: 18 22 18
  </Values>
</DailyCycle>
```

#### 14.3.4. Vermeidung von Sprüngen / Leistungsverbesserung

Bei der Definition von Tageszyklen mit dem Interpolationsmodus **Constant** springen die Werte tatsächlich zwischen den Intervallen. Diese Diskontinuitäten sind sehr teuer in der Berechnung, da der Solver Zeitschritte um diese Sprünge herum gruppieren muss, um den Schrittfunktionen genau zu folgen.

Für praktische Anwendungen sind diese Schritte jedoch oft nicht erwünscht - auch wenn ein Sollwert kurzzeitig auf einen neuen Wert umgeschaltet wird, kann es in der Tat einige Minuten dauern, bis der resultierende physikalische Effekt spürbar wird. Dies wird bei der Interpretation der Sollwerte durch den Solver berücksichtigt.

Anstatt exakt die schrittweise geplanten Werte zu liefern, implementiert der Solver eine automatische 2-Minuten-Rampe kurz vor dem Intervallende. [Abbildung 9](#) veranschaulicht die 2-minütige lineare Rampe, die direkt vor jedem neuen Intervall angewendet wird.

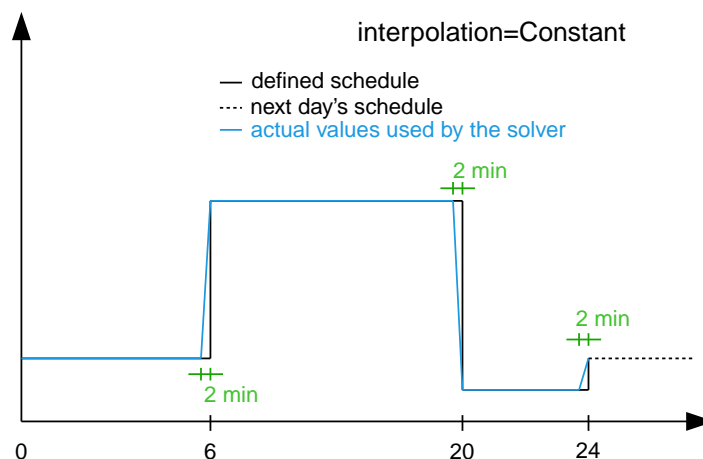


Abbildung 9. Rampen-/Schrittglättung angewandt auf Tageszyklen mit schrittweise definierten Werten

!

Der Rampenzeitabstand von 2 Minuten ist derzeit in der Zeitplan-Berechnungsroutine fest codiert und kann bei Bedarf auf einen größeren oder kleineren Wert geändert werden. Außerdem kann anstelle einer linearen Rampenfunktion eine polynomische Kurve 3. Ordnung verwendet werden (was immer der bester Kompromiss zwischen



Leistung und Genauigkeit ist).

\$

Intern wird die Schrittglättung realisiert, indem 2 Minuten vor dem Intervallende ein neuer Datenpunkt mit dem gleichen Wert wie im aktuellen Intervall eingefügt wird. Der Tageszyklus wird dann wie ein linear interpolierter Tageszyklus behandelt. Es gibt jedoch keine Prüfung für Intervalllängen kleiner als 2 Minuten. Daher müssen bei der Definition von Tageszyklen mit Interpolationsmodus **Linear** keine Intervalle kleiner oder gleich 2 Minuten definiert werden.

## 14.4. Jahresschaltpläne

Jahrespläne sind im Grunde Datentabellen mit monoton ansteigenden X (Zeit)-Werten. Jeder Jahresplan definiert eine einzelne Größe. Es können z. B. stündliche Werte von Temperaturen oder Steuergrößen angegeben werden, die während des Jahres gemessen werden.

!

Der Name *annual schedule* ist eigentlich etwas irreführend. In diesen Datentabellen können Sie Daten mit beliebigen Zeitspannen unterbringen, die nur wenige Wochen oder sogar mehrere Jahre umfassen (z. B. mit Überwachungsdaten). Die einzige Voraussetzung ist, dass das Zeitintervall der Simulation in die Zeitspanne des Zeitplans passt.

Die vom linearen Spline gelieferten Werte können als linear/konstant interpolierte Werte definiert werden, allerdings sollte aus Performance-Gründen der konstante Interpolationsmodus vermieden werden.

!

Bei linearen Splines wird die Schrittglättung nicht vom Solver angewendet. Es ist Sache des Anwenders, geeignete Daten bereitzustellen oder durch langsame Simulationszeiten bestraft zu werden.

### 14.4.1. Definition von Jahreszeitplänen im XML-File

Innerhalb des XML-tags **Annual Schedules** gibt es ein oder mehrere XML-Tochtertags **ScheduleGroup**, jedes mit einem obligatorischen XML-Attribut **objectList**. Dieses referenziert, genau wie bei den täglichen Zyklusplänen eine Objektliste und damit Objekte, auf die sich die geplanten Variablen beziehen. [Beispiel 37](#) zeigt ein Beispiel für jährliche Zeitpläne, die innerhalb einer einzigen **ScheduleGroup** definiert sind.

*Beispiel 37. Definition von Jahresplänen*

```
<Annual Schedules>
  ...
  <ScheduleGroup objectList="All zones">
    <Annual Schedule name="HeatingSetPointTemperature" interpolation="linear">
      <X unit="h"> 0 2183 2184 6576 6577 8760 </X>
      <Y unit="C"> 20 30 20 30 20 30 20 30 </Y>
    </Annual Schedule>
    <Annual Schedule name="TotalEnergyProductionPerPerson" interpolation="linear">
      <X unit="h"> 0 2183 2184 6576 6577 8760 </X>
      <Y unit="W/Person"> 70 110 70 110 70 110 </Y>
    </Annual Schedule>
  </ScheduleGroup>
</Annual Schedules>
```

```

&#x2013; </AnnualSchedule>
&#x2013; <AnnualSchedule name="EquipmentUtilizationRatio" interpolation="linear">
&#x2013; <X unit="h"> 0 2183 2184 6576 6577 8760</X>
&#x2013; <Y unit="W/Person"> 10 20 10 20 10 20 </Y>
&#x2013; </AnnualSchedule>
&#x2013; </ScheduleGroup>
&#x2013; ...
&#x2013; </AnnualSchedules>

```

Die eigentlichen Daten werden in den XML-tags **AnnualSchedule** angegeben, die eigentlich ein **LinearSplineParameter** sind (siehe referenzierte Dokumentation für Details).

Die Einheit des X-Wertes muss eine Zeiteinheit sein. Die Einheit des Y-Wertes ist die Einheit der geplanten Menge.

#### 14.4.2. Definition von Jahreszeitplänen durch Einbindung von TSV-Dateien

Ein Jahreszeitplan kann ebenso über die Einbindung von Daten aus einem \*.tsv-File eingebunden werden. Hierbei muss die Datei den folgenden Konventionen entsprechen:

- ⌘ Die Nullte Spalte enthält das Zeitintervall. Deren Einheit ist frei wählbar (a,d,h,min,s).
- ⌘ Die nachfolgenden Spalten müssen mit der jeweiligen Einheit des Parameters im Header versehen werden.

##### Beispiel 38. Beispiel tsv-Datei

```

Time [h]   PersonHeatLoadPerAreaSchedule [W/m2]   EquipmentHeatLoadPerAreaSchedule [W/m2]
0   10   30
1   15   30
2   20   30
3   10   30
4   20   30
...

```

##### Beispiel 39. Definition von Jahresplänen aus TSV-Datei

```

<AnnualSchedules>
&#x2013; ...
&#x2013; <ScheduleGroup objectList="All zones">
&#x2013; <AnnualSchedule name="PersonHeatLoadPerAreaSchedule" interpolationMethod="constant">
&#x2013; <TSVFile>$(Project Directory)/Schedules.tsv?1</TSVFile>
&#x2013; </AnnualSchedule>
&#x2013; </ScheduleGroup>
&#x2013; ...
&#x2013; </AnnualSchedules>

```



Die Angabe der Spalte des zeitplangesteuerten Parameters aus dem TSV-File wird am Ende des Dateipfades festgelegt.

## 14.5. Variablenliste

Die Variablenliste beschreibt alle Namen und die Einheiten, die in den Zeitplänen verwendet werden können.

Tabelle 21. Variablenliste

Name	Einheit	Beschreibung
HeatingSetPointTemperature	C	Sollwerttemperatur für Heizung
CoolingSetPointTemperature	C	Sollwerttemperatur für Kühlen
AirConditionSetPointTemperature	C	Solltemperatur für die Klimatisierung
AirConditionSetPointRelativeHumidity	%	Sollwert der relativen Luftfeuchtigkeit für die Klimatisierung
AirConditionSetPointMassFlux	kg/s	Sollwert Massenstrom für die Klimatisierung
HeatingLoad	W	Heizlast
Thermal Load	W	Thermische Last (positiv oder negativ)
MoistureLoad	g/h	Feuchtelast
CoolingPower	W	Kühlleistung
LightingPower	W	Beleuchtungsleistung
Thermal EnergyLossPerPerson	W/Person	Energie der Aktivitäten einer einzelnen Person, die nicht als Heizwärme zur Verfügung steht
Total EnergyProductionPerPerson	W/Person	Gesamtenergieproduktion des Körpers einer einzelnen Person bei einer bestimmten Tätigkeit
MoistureReleasePerPerson	kg/s	Feuchtigkeitsabgabe eines einzelnen Personenkörpers bei einer bestimmten Tätigkeit
CO2EmissionPerPerson	kg/s	CO <sub>2</sub> -Emissionsmassenstrom einer einzelnen Person bei einer bestimmten Tätigkeit
MassFluxRate	---	Fraktion des realen Massenstroms zum maximalen Massenstrom für verschiedene Tageszeiten
PersonHeatLoadPerAreaSchedule	W/m <sup>2</sup>	Interne Wärmelast infolge flächenabhängiger Personenbelegung
PressureHead	Pa	Versorgungsdruckhöhe einer Pumpe
OccupancyRate	---	Fraktion der realen Belegung zur maximalen Belegung für verschiedene Tageszeiten
EquipmentUtilizationRatio	---	Verhältnis des Verbrauchs für vorhandene elektrische Geräte
LightingUtilizationRatio	---	Verhältnis des Verbrauchs für die Beleuchtung

Name	Einheit	Beschreibung
MaximumSolarRadiationIntensity	W/m <sup>2</sup>	Maximale Sonneneinstrahlungsintensität bevor die Beschattung aktiviert wird
UserVentilationAirChangeRate	1/h	Austauschrate für natürliche Lüftung
UserVentilationComfortAirChangeRate	1/h	Maximale Luftwechselrate = Offset für Nutzerkomfort
UserVentilationMinimumRoomTemperature	°C	Temperaturgrenze, ab der die Komfortlüftung aktiviert wird
UserVentilationMaximumRoomTemperature	°C	Temperaturgrenzwert, unterhalb dessen die Komfortlüftung aktiviert wird
InfiltrationAirChangeRate	1/h	Austauschrate für Infiltration
ShadingFactor	---	Schattierungsfaktor [0 ÷ 1]

## 15. Outputs/Ergebnisse

In NANDRAD ist es möglich, Ausgabedaten für jede berechnete und veröffentlichte Größe abzurufen (siehe [Mengenreferenzen](#) für eine vollständige Liste). Natürlich sind nicht alle Größen in allen Projekten verfügbar - vieles hängt davon ab, welche Art von Modellen und Geometrien definiert werden.

Um eine Ausgabe zu definieren, werden die folgenden Informationen benötigt:

- ein Ausgabemaster, definiert *wann* Ausgaben geschrieben werden sollen
- den Variablennamen (Bezeichner für die physikalische Größe)
- eine Objektliste, die das Objekt oder die Objekte auswählt, von denen Daten abgerufen werden sollen
- (optional) Informationen zur Zeitbehandlung, d. h. ob ein zeitlicher Mittelwert gebildet oder eine Zeitintegration durchgeführt werden soll
- (optional) Zieldateiname

Zusätzlich zu den manuell definierten Ausgaben erzeugt NANDRAD auch automatisch eine Reihe von Log- und Datendateien (siehe Abschnitt [Solver-Logdateien](#)).

Die Ausgaben werden im XML-tag **Outputs** definiert, mit der folgenden allgemeinen Struktur:

*Beispiel 40. Parameter-Definition für Ausgaben*

```

<Outputs>
  ... <!-- globale Ausgabeparameter -->

  <Grids>
    ... <!-- Definition von Ausgabemastern -->
  </Grids>

  <Definitions>
    ... <!-- Tatsächliche Ausgangsdefinitionen -->

```

```
É </Definitions>
</Outputs>
```

## 15.1. Globale Ausgabeparameter

Die folgenden Parameter beeinflussen die Erzeugung der Ausgabedateien:

- ¥ **TimeUnit** - der Wert dieses XML-tags enthält die Zeiteinheit, die in den Ausgabedateien verwendet werden soll (nur bei Dateien im ASCII-Format)
- ¥ **IBK:Flag** - namens **BinaryFormat**: falls wahr, werden die Dateien im Binärformat geschrieben (siehe **Binäres Format**).

*Beispiel 41. Globale Ausgabeparameter*

```
<Outputs>
É <TimeUnit>d</TimeUnit>
É <IBK:Flag name="BinaryFormat">false</IBK:Flag>
É ...
</Outputs>
```

## 15.2. Ausgaberaaster

Ausgaberaaster legen fest, *wann* Ausgaben geschrieben werden. Ein Ausgaberaaster enthält eine Liste von Intervallen, wobei für jedes Intervall eine Ausgabenschrittgröße definiert ist. Wenn Sie z. B. stündliche Ausgabeschritte von Anfang bis Ende haben möchten, müssen Sie ein Raster mit einem Intervall und einem Schrittgrößenparameter von einer Stunde definieren:

*Beispiel 42. Ausgaberaaster für die gesamte Simulation mit stündlichen Schritten*

```
<Grids>
É <OutputGrid name="hourly">
É <Intervals>
É <Interval>
É <IBK:Parameter name="StepSize" unit="h">1</IBK:Parameter>
É </Interval>
É </Intervals>
É </OutputGrid>
</Grids>
```

Ein Ausgaberaaster wird durch seinen Namen (obligatorisches XML-Attribut **name**) eindeutig identifiziert. Es enthält ein einzelnes untergeordnetes XML-tag **Intervals**, das ein oder mehrere Intervalle enthält. Es wird erwartet, dass die Intervalle (XML-tag **Interval**) zeitlich aufeinander folgen. Es sind Lücken dazwischen möglich.

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Start	h	die Startzeit des Intervalls (siehe Erläuterung unten) der Wand	$\geq 0.0$	optional/erforderlich
End	h	die Endzeit des Intervalls (siehe Erläuterung unten)	$\geq 0.0$	optional/erforderlich
StepSize	h	der Abstand zwischen den Ausgaben innerhalb des Intervalls	$> 0.0$	erforderlich

Die Parameter werden in XML-tags vom Typ `IBK:Parameter` gespeichert, siehe `IBK:Parameter`.

Die Zeitpunkte in den Parametern `Start` und `End` werden in Bezug auf Mitternacht des 1. Januar des jeweiligen Jahres definiert, in dem die Simulation beginnt.

### 15.2.1. Regeln

Der Parameter `Start` ist unter den folgenden Bedingungen optional:

- # im ersten Intervall wird ein fehlender `Start`-Parameter automatisch auf 0 gesetzt (Beginn des Jahres)
- # in allen anderen Intervallen wird die `End`-Zeit des vorangegangenen Intervalls genommen (siehe folgende Regel)

Die Endzeit eines Intervalls wird definiert, entweder:

- # durch Definition des Parameters `End`,
- # durch Definition des Parameters `Start` im nächsten Intervall
- # durch die Endzeit der Simulation (nur im letzten Intervall)

Grundsätzlich muss für den Solver klar sein, wann ein Intervall beginnt und endet, und wie groß die Schrittweite ist.

Während der Simulation wird eine Ausgabe genau unter der folgenden Bedingung geschrieben:

- Der Offset  $t$  muss innerhalb eines durch das Gitter definierten Intervalls liegen
- Der Offset  $t$  vom Beginn des Intervalls muss ein exaktes Vielfaches der Schrittweite sein

#### Beispiel 43. Ausgaberauswertung

Angenommen, ein Ausgabeintervall ist so definiert, dass es bei 12,5 h beginnt, mit einer Schrittweite von 2 h. Die Simulationszeit soll  $t=16,5$  h betragen. Dann wäre  $16,5 - 12,5 = 4$  h. 4h ist ein exaktes Vielfaches von 2 h. Das Ausgaberaaster wäre also zu diesem Simulationszeitpunkt "aktiv" und alle Ausgaben, die mit diesem Ausgangsgitter verbunden sind, werden geschrieben.

Zwischen den Intervallen kann es Lücken geben, in denen keine Ausgaben geschrieben werden:

*Beispiel 44. Ausgaberaaster für Tageswerte im ersten Jahr und Stundenwerte im dritten Jahr (beginnend zum Zeitpunkt "2 a")*

```
<Grids>
  <OutputGrid name="first_and_last">
    <Intervals>
      <Interval>
        <IBK:Parameter name="StepSize" unit="d">1</IBK:Parameter>
        <IBK:Parameter name="End" unit="a">1</IBK:Parameter>
      </Interval>
      <Interval>
        <IBK:Parameter name="Start" unit="a">2</IBK:Parameter>
        <IBK:Parameter name="StepSize" unit="h">1</IBK:Parameter>
      </Interval>
    </Intervals>
  </OutputGrid>
</Grids>
```

## 15.3. Ausgangsdefinitionen

Nachfolgend finden Sie ein Beispiel für eine Ausgabedefinition:

*Beispiel 45. Ausgabe der Lufttemperatur von allen Zonen in der Objektliste All zones und unter Verwendung des Ausgaberasters hourly*

```
<Definitions>
  <OutputDefinition>
    <Quantity>AirTemperature</Quantity>
    <ObjectName>All zones</ObjectName>
    <GridName>hourly</GridName>
  </OutputDefinition>
  ... <!-- weitere Definitionen -->
</Definitions>
```

*Beispiel 46. Ausgabe der Globalstrahlung auf eine Fläche eines bestimmten Sensor id=2000000 von der Location in der Objektliste Location und unter Verwendung des Ausgaberasters hourly sowie einer separaten Ausgabedatei Sensordata.tsv*

```
<Definitions>
  <OutputDefinition>
    <Quantity>GlobalSWRadOnPlane[2000000]</Quantity>
    <ObjectName>Location</ObjectName>
    <GridName>hourly</GridName>
    <!-- die Globalstrahlungsdaten werden in der Datei Sensordata.tsv separat abgelegt -->
    <FileName>Sensordata<FileName>
  </OutputDefinition>
  ... <!-- weitere Definitionen -->
</Definitions>
```

Das Beispiel zeigt die obligatorischen Elemente des XML-tags **OutputDefinition**. Im Folgenden finden Sie eine Liste aller unterstützten Elemente:

XML-tag	Beschreibung	Verwendung
<b>Quantity</b>	Eindeutiger ID-Name der physikalischen Größe, siehe auch <a href="#">Mengenreferenzen</a>	<i>erforderlich</i>
<b>ObjectListName</b>	Referenz auf eine Objektliste, die die Objekte identifiziert von denen Ergebnisse genommen werden sollen	<i>erforderlich</i>
<b>GridName</b>	Referenz auf ein Ausgabegitter (Ausgabezeitdefinitionen)	<i>erforderlich</i>
<b>FileName</b>	Zielfilename	<i>optional</i>
<b>TimeType</b>	Methode der Zeitmittelung/Integration	<i>optional</i>

Der ID-Name der Ergebnisgröße ist der Name des Ergebnisses eines Modellobjekts, eines Zeitplans oder eines anderen vom Solver erzeugten Objekts. Das entsprechende Objekt oder die entsprechenden Objekte werden durch eine [Objektliste](#) ausgewählt. Der Gittername ist der ID-Name eines [Ausgabegitters](#).

Das Element **FileName** ist optional. Er kann verwendet werden, um gezielt den Namen einer Ausgabedatei auszuwählen. Normalerweise werden die Namen der Ausgabedateien automatisch generiert, abhängig von der Art der angeforderten Ausgabe.

Schließlich kann das Element **TimeType** verwendet werden, um die zeitliche Mittelung oder die zeitliche Integration von Variablen festzulegen, siehe Abschnitt [Zeittypen](#).

### 15.3.1. Variablennamen und Variablennachschlageregeln

Mengen in Ausgabedefinitionen definieren die ID-Namen der Ausgabegrößen. Wenn ein Element einer vektoriellen Größe angefordert wird, muss das betreffende Element über eine Index-Notation definiert werden. Dabei sind die folgenden Notationen erlaubt:

- ¥ **HeatSource[1]** - das Index-Argument wird so interpretiert, wie es von den bereitstellenden Modellen definiert wird, wenn also das Modell eine vektorwertige Größe mit Modell-ID-Indizierung bereitstellt, wird das Argument als Objekt-ID interpretiert (ansonsten als Positionsindex)
- ¥ **HeatSource[index=1]** - das Argument index wird explizit als Positionsindex interpretiert (führt zu einem Fehler, wenn das Modell eine Größe mit Modell-ID-Indizierung bereitstellt)
- ¥ **HeatSource[id=1]** - das index-Argument wird explizit als Objekt-ID interpretiert (führt zu einem Fehler, wenn das Modell eine Menge mit Positionsindizierung liefert)

### 15.3.2. Ausgabefilenames

Die folgenden Abschnitte beschreiben die Regeln, die die Ausgabefilenames bestimmen.

Wenn kein Dateiname angegeben wird

Zielfilename(n) werden automatisch festgelegt.



Alle Ausgaben werden abhängig von der physikalischen Größe gruppiert in:

- ¥ Zustände : *states*
- ¥ Ströme : *fluxes*
- ¥ Lasten : *load*
- ¥ Sonstiges : *misc*

Wenn **Integral** als **TimeType** gewählt wird:

- ¥ Für Ausgaben vom Typ *fluxes* wird stattdessen die Gruppe *flux\_integrals* verwendet,
- ¥ Für Ausgaben vom Typ *loads* wird stattdessen die Gruppe *load\_integrals* verwendet

Die Ausgaben werden weiter nach dem Namen des Ausgaberasters gruppiert. Der endgültige Ausgabedateiname wird für jeden Gitter- und Gruppennamen ermittelt:

- ¥ *states* ! *states\_<gridname>.tsv*
- ¥ *loads* ! *loads\_<gridname>.tsv*
- ¥ *loads (integriert)* ! *load\_integrals\_<gridname>.tsv*
- ¥ *fluxes* ! *fluxes\_<gridname>.tsv*
- ¥ *fluxes (integriert)* ! *flux\_integrals\_<gridname>.tsv*

!

Es gibt eine Sonderregel: Wenn nur ein Gitter verwendet wird, wird das Suffix *\_<gridname>* weggelassen.

Wenn ein Dateiname angegeben wird

Die Menge wird in die angegebene Datei geschrieben. Wenn es mehrere Ausgabedefinitionen mit demselben Dateinamen gibt, werden alle Mengen in dieselbe Datei geschrieben, unabhängig vom Typ.

||

Alle Ausgabedefinitionen mit demselben Dateinamen müssen das gleiche Raster verwenden (gleiche Zeitpunkte für alle Spalten sind erforderlich!)

### 15.3.3. Zeittypen

Das tag **TimeType** nimmt die folgenden Werte an:

- ¥ **None** - schreibt die Ausgaben wie zum Ausgabezeitpunkt errechnet
- ¥ **Mean** - schreibt den über das letzte Ausgabeintervall gemittelten Wert
- ¥ **Integral** - schreibt das Zeitintegral der Ergebnisgröße (Integration beginnt zu Simulationsbeginn stets bei 0)

Standardmäßig (wenn das Element **TimeType** nicht explizit angegeben ist) werden die Werte so geschrieben, wie sie zum Ausgabezeitpunkt berechnet werden (entspricht **None**). Abbildung [Illustration der verschiedenen TimeType-Optionen](#) veranschaulicht die verschiedenen Optionen.

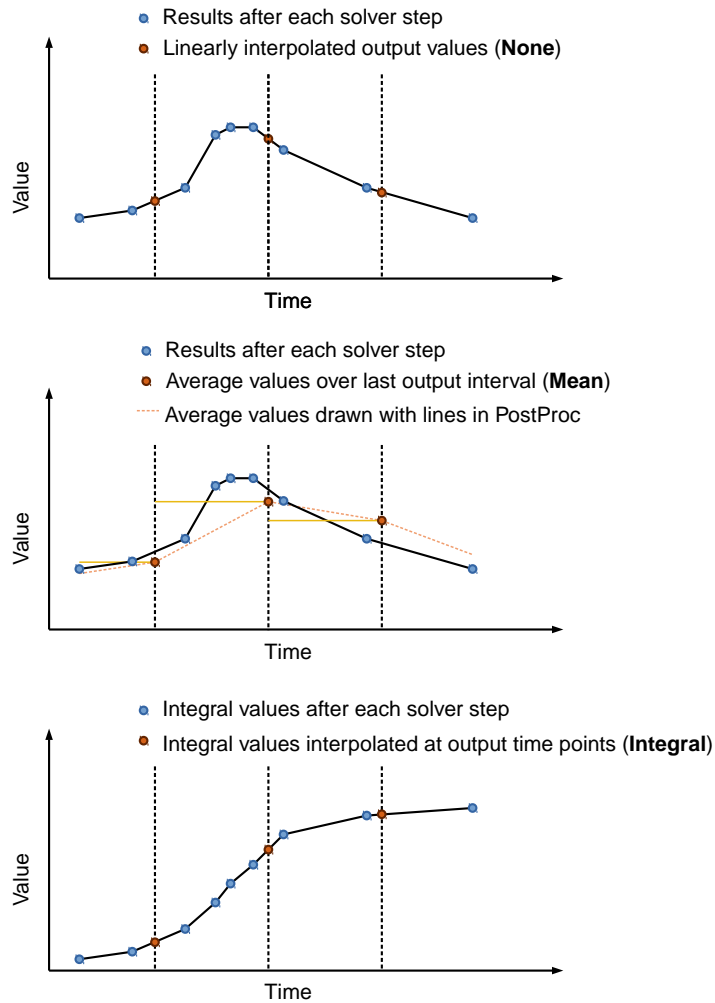


Abbildung 10. Illustration der verschiedenen *TimeType*-Optionen

■

Es ist wichtig zu beachten, dass Durchschnittswerte immer Mittelwerte der Werte im letzten Ausgabeintervall sind. Wenn Sie also stündliche Ausgänge definiert haben, aber die Einheit *kW/d* ist, erhalten Sie keine Durchschnittswerte über einen Tag, sondern über die letzte Stunde. Die Einheit wird nur zur Umrechnung des Endwertes benötigt, hat aber keinen Einfluss auf die Art der Berechnung.

#### 15.3.4. Beispiele

##### Beispiel 47. Abfrage von Oberflächentemperaturen der Konstruktionen

```
<Outputs>
E ...
E <Definitions>
E   <OutputDefinition>
E     <Quantity>SurfaceTemperatureA</Quantity>
```

```

&E      <ObjectName>Walls</ObjectName>
&E      <GridName>hourly</GridName>
&E      </OutputDefinition>
&E      <OutputDefinition>
&E      <Quantity>SurfaceTemperatureB</Quantity>
&E      <ObjectName>Walls</ObjectName>
&E      <GridName>hourly</GridName>
&E      </OutputDefinition>
&E      ... <!-- weitere Definitionen -->
&E      </Definitions>
</Outputs>
<ObjectLists>
&E <ObjectList name="Walls">
&E   <FilterID>*</FilterID>
&E   <!-- Objektliste muss auf Konstruktionsinstanzen verweisen -->
&E   <ReferenceType>ConstructionInstance</ReferenceType>
&E </ObjectList>
&E   ... <!-- andere Objektlisten -->
</ObjectLists>

```

*Beispiel 48. Anforderung von Energie, die der Schicht in einer Konstruktion zugeführt wird (Fußbodenheizung)*

```

<Outputs>
&E   ...
&E   <Definitions>
&E       <OutputDefinition>
&E           <!-- Index 1 = Wärmequelle in Schicht 1, von Seite A aus zählend -->
&E           <Quantity>HeatSource[1]</Quantity>
&E           <ObjectName>FloorHeating1</ObjectName>
&E           <GridName>hourly</GridName>
&E       </OutputDefinition>
&E       ... <!-- weitere Definitionen -->
&E   </Definitions>
</Outputs>
<ObjectLists>
&E <ObjectList name="FloorHeating1">
&E   <FilterID>15</FilterID>
&E   <!-- Objektliste muss Bauinstanzen referenzieren -->
&E   <ReferenceType>ConstructionInstance</ReferenceType>
&E </ObjectList>
&E   ... <!-- andere Objektlisten -->
</ObjectLists>

```

## 15.4. BinSres Format

Falls der Schalter **BinaryFormat** eingeschaltet ist, werden die Ergebnisse als **btf** Dateien (*binary table format*) geschrieben. Dieses Format wird nativ von *PostProc* unterstützt und die Daten können genau wie **tsv**-Dateien eingelesen werden.

Das binSre Dateiformat ist im *PostProc*-Handbuch beschrieben:

<https://bauklimatik-dresden.de/postproc/help/de/index.html#binaryFormat>

# 15.5. Solver-Logdateien

Innerhalb des Ergebnisverzeichnis des Projekts werden automatisch die folgenden Dateien erzeugt:

```
! " " log
# ! " " integrator_ccode_stats.tsv
# ! " " LES_direct_stats.tsv
# ! " " progress.tsv
# ! " " screenlog.txt
# $ " " summary.txt
! " " results
# $ " " ... (output files)
$ " " var
Ê ! " " input_reference_list.txt
Ê ! " " objectref_substitutions.txt
Ê ! " " output_reference_list.txt
Ê $ " " restart.bin
```

Datei	Beschreibung
integrator_ccode_stats. tsv	Statistik des Zeitintegrators, wird am Ende der Simulation geschrieben
LES_direct_stats. tsv	Statistik des Linear Equation System (LES) Solvers, wird am Ende der Simulation geschrieben
progress. tsv	Minimalistische Laufzeit-Fortschrittsdaten, kontinuierlich geschrieben, kann zum Verfolgen des Simulationsfortschritts vom GUI-Tool verwendet werden
screenl og.txt	Log-Datei fŸr Solver-Ausgabemeldungen (wie Konsolenfensterausgaben), wird kontinuierlich geschrieben
summary. txt	Statistiken und Zeitangaben des Simulationslaufs, wird am Ende der Simulation geschrieben
input_reference_list. txt	Liste der von Modellen verwendete EingangsgröŸen (ErgebnisgröŸen anderer Modelle) (siehe <a href="#">Mengenreferenzen</a> )
output_reference_list. txt	Liste der in diesem Projekt erzeugten GröŸen (siehe <a href="#">Mengenreferenzen</a> )
objectref_substi tutions.txt	Liste von Objektreferenzen (einschlieŸlich IDs), wie sie in Ausgabedateien erscheinen und deren <i>Displayname</i> Attributen (wenn vergeben). Kann benutzt werden, um die generischen Bezeichner in lesbare Begriffe zu Ÿbersetzen.
restart. bin	Binäre Neustartdaten (zur Fortsetzung der Integration/des Solvers)



Wenn Sie einen anderen Integrator oder Solver fŸr lineare Gleichungssysteme gewŸhlt haben (siehe Abschnitt [Solver-Parameter](#)), werden die Dateien `integrator_ccode_stats.tsv` und `LES_direct_stats.tsv` entsprechend anders benannt.

# 16. Globale Parameter

Durch die globalen Simulationsoptionen werden folgende Punkte gesteuert:

- wie das Modell arbeitet
- die Berechnungsgenauigkeit (wirkt sich auf die Leistung aus)
- die Berechnungsleistung

Die einzelnen Einstellungen sind aufgeteilt in *Simulationparameter* und *Solver-Parameter*, wobei sich letztere auf das numerische Lösungsverfahren beziehen.

## 16.1. Simulationsparameter

Im Folgenden werden alle Simulationsparameter beschrieben, siehe [Beispiel 49](#). Alle Parameter werden als `IBK:Parameter`, `IBK:Flags` oder `IBK:IntPara` gesetzt.

Beispiel 49. *Simulationparameter*

```
<SimulationParameter>
  <IBK:Parameter name="Initial Temperature" unit="C">5</IBK:Parameter>
  <Interval>
    <IBK:Parameter name="Start" unit="d">0</IBK:Parameter>
    <IBK:Parameter name="End" unit="d">730</IBK:Parameter>
  </Interval>
</SimulationParameter>
```

Das tag `SimulationParameter` enthält die folgenden untergeordneten tags:

XML-tag	Beschreibung	Verwendung
<code>IBK:Parameter</code>	Schwebepunktwertparameter	mehrfach
<code>IBK:IntPara</code>	Ganzzahlige Parameter	vielfach
<code>IBK:Flag</code>	Flags	vielfach
<code>Interval</code>	Definiert das Simulationsintervall	kein/einmal

Fließkommaparameter (siehe Abschnitt [IBK:Parameter](#) für eine Beschreibung des tags `IBK:Parameter`):

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
<code>Initial Temperature</code>	C	Globale Anfangstemperatur für alle Objekte ( <code>Zonen</code> , <code>Baueinstanzen</code> , etc)	positive double (0.0K)	<i>optional</i>

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
(*)InitialRelativeHumidity	%	Globale anfängliche relative Luftfeuchtigkeit für alle Objekte, die einen Feuchtwert gesetzt haben können (Zonen, Luftströme in Modellen, etc)	0 ÷ 100%	optional
(*)RadiationLoadFraction	---	Prozentualer Anteil der solaren Strahlungsgewinne, die direkt dem Raum zugerechnet werden 0..1.	0 ÷ 1	optional
(*)UserThermalRadiationFraction	---	Prozentualer Anteil der Wärme, die durch langwellige Strahlung von Personen abgegeben wird.	0 ÷ 1	optional
(*)EquipmentThermalLossFraction	---	Prozentualer Anteil der Energie aus der Gerätebelastung, der nicht als thermische Wärme zur Verfügung steht.	0 ÷ 1	optional
(*)EquipmentThermalRadiationFraction	---	Prozentualer Anteil der Wärme, die durch langwellige Strahlung von Geräten abgegeben wird.	0 ÷ 1	optional
(*)LightingVisibleRadiationFraction	---	Prozentualer Anteil der Energie der Beleuchtung, die in sichtbare kurzwellige Strahlung umgewandelt wird.	0 ÷ 1	optional
(*)LightingThermalRadiationFraction	---	Prozentualer Anteil der Energie der Beleuchtung, die in langwellige Strahlung umgesetzt wird.	0 ÷ 1	optional
(*)DomesticWaterSensitiveHeatGainFraction	---	Prozentualer Anteil der sensiblen Wärme des Brauchwassers, der an den Raum abgegeben wird.	0 ÷ 1	optional
(*)AirExchangeRateN50	1/h	Luftwechselrate, die sich aus einer Druckdifferenz von 50 Pa zwischen innen und außen ergibt.	positive double (≥0.0)	optional
(*)ShieldingCoefficient	---	Abschirmkoeffizient für einen bestimmten Ort und Hüllentyp.	0 ÷ 1	optional
(*)HeatingDesignAmbientTemperature	C	Umgebungstemperatur für einen Auslegungstag. Parameter, der für den FMU-Export benötigt wird.	positive double (≥0.0)	optional

(\*) - bisher noch nicht verwendet

Ganzzahlige Parameter (siehe Abschnitt [IBK: IntPara](#) für eine Beschreibung des tags [IBK: IntPara](#)):

Name	Beschreibung	Standard	Verwendung
StartJahr	Startjahr der Simulation	2001	optional

Flags und Optionen (siehe Abschnitt [IBK:Flag](#) für eine Beschreibung des tags [IBK:Flag](#)):

Name	Beschreibung	Standard	Verwendung
(*)EnableMoistureBalance	Flag, das die Berechnung der Feuchtigkeitsbilanz aktiviert, wenn diese aktiviert ist	false	optional
(*)EnableCO2Balance	Flag, das die Berechnung der CO2-Bilanz aktiviert, wenn aktiviert	false	optional
(*)EnableJointVentilation	Flag, das die Belüftung durch Fugen und ...ffnungen aktiviert.	false	optional
(*)ExportClimateDataFMU	Flag, die den FMU-Export von Klimadaten aktiviert.	false	optional

(\*) - bisher noch nicht verwendet

### 16.1.1. Simulationszeitintervall

Der tag [SimulationParameters](#) enthält auch den Start und das Ende der Simulation. Standardmäßig ist das Simulationszeitintervall so eingestellt, dass es sich über ein ganzes Jahr erstreckt, beginnend um Mitternacht am 1. Januar. Es ist jedoch möglich, ein anderes Zeitintervall zu definieren und damit auch eine Simulation, die länger als ein Jahr läuft.

Dies wird im untergeordneten tag [Interval](#) gemacht:

Das Simulationsintervall beginnt am 1. Februar (kurz nachdem die ersten 31 Tage des Januars vorbei sind) und läuft 60 Tage.

```
<Interval>
  <IBK:Parameter name="Start" unit="d">31</IBK:Parameter>
  <IBK:Parameter name="End" unit="d">91</IBK:Parameter>
</Interval>
```

Der Start und das Ende einer Simulation werden immer in *simulation time* definiert, was im nächsten Abschnitt genauer erklärt wird.

### 16.1.2. Simulationszeit und absoluter Zeitbezug

NANDRAD verwendet zwei Zeitmaße:

- Simulationszeit, die beim Start der Simulation immer bei 0 beginnt, und

¥ Absolute Zeit, die die in ein reales Datum/Uhrzeit umgerechnete Zeit ist und auf dem tatsächlichen Startzeitpunkt der Simulation basiert.

Die *Simulationszeit* beschreibt grundsätzlich einen Zeitversatz relativ zum Startpunkt der Simulation und wird typischerweise nur als Zeitdelta ausgedrückt, z. B. "20 d" oder "15.5 h".

Die *Absolute Zeit* ist eine bestimmte Zeit/ein bestimmtes Datum, z. B. "20.09.2020 14:30", die/der sich durch Addition des Offsets der *Simulationszeit* zu einem Startzeitpunkt ergibt.

In NANDRAD wird dieser Simulationsstartzeitpunkt in zwei Parametern angegeben:

¥ das **StartYear** und

¥ das Offset der Zeit seit Beginn (Mitternacht 1. Januar) dieses Jahres als **Start** Intervallparameter.

Ein **Start**-Offset von **1 d** lässt die Simulation am *Januar 2, 0:00* beginnen. Wenn die Simulation z.B. am *15. Januar 2003, 6:00* beginnen soll, muss folgendes angegeben werden:

```
StartYear = 2003
Start = 14*24 + 6 = 342 h
```

Und für den letzten Tag des Jahres muss die Simulation bei **Start = 364 d** gestartet werden.

!!

In NANDRAD gibt es keine Schaltjahre. Selbst wenn Sie 2004 als Startjahr angeben, wird es keinen 29. Februar geben! Wenn Sie eine Mehrjahressimulation durchführen, hat jedes Jahr 365 Tage.

## 16.2. Solver-Parameter

Im Folgenden werden alle Parameter beschrieben, die für den Solver benötigt werden.

Solver-Parameter

```
<SolverParameter>
  <IBK:Parameter name="MaxTimeStep" unit="min">30</IBK:Parameter>
  <IBK:Parameter name="MinTimeStep" unit="s">1e-4</IBK:Parameter>
  <IBK:Parameter name="Rel Tol" unit="---">1e-005</IBK:Parameter>
  <IBK:Parameter name="AbsTol" unit="---">1e-006</IBK:Parameter>
  <IBK:Parameter name="NonLinSolverConvCoeff" unit="---">1e-05</IBK:Parameter>
  <IBK:IntPara name="MaxKrylovDim">30</IBK:IntPara>
  <IBK:Parameter name="DiscMinDx" unit="mm">2</IBK:Parameter>
  <IBK:Parameter name="DiscStretchFactor" unit="---">4</IBK:Parameter>
  <IBK:IntPara name="DiscMaxElementsPerLayer">30</IBK:IntPara>
  <IBK:Flag name="DetectMaxTimeStep">true</IBK:Flag>
  <Integrator>CVODE</Integrator>
  <LesSolver>Dense</LesSolver>
</SolverParameter>
```

Der tag **SolverParameter** enthält die folgenden untergeordneten Elemente:



XML-tag	Beschreibung	Verwendung
<b>IBK: Parameter</b>	Parameter für Fließkommazahlen	mehrfach
<b>IBK: IntPara</b>	Ganzzahlige Parameter	vielfach
<b>IBK: Flag</b>	Flags	mehrfach
<b>Integrator</b>	Definiert Zeitintegrator	kein/einmal
<b>LesSolver</b>	Definiert Solver für lineare Gleichungssysteme (LES)	kein/einmal
<b>Preconditioner</b>	Definiert Vorkonditionierer (nur iterativer LES-Solver)	einzeln/einmal

Fließkommaparameter (siehe Abschnitt **IBK:Parameter** für eine Beschreibung des tags **IBK: Parameter**):

Name	Vorgabe Einheit	Beschreibung	Wertebereich	Vorgabe	Verwendung
<b>Rel Tol</b>	---	Relative Toleranz für die Fehlerprüfung des Solvers.	$0 \leq 0.1$	1E-04	<i>optional</i>
<b>AbsTol</b>	---	Absolute Toleranz für die Fehlerprüfung des Solvers.	$0 \leq 1$	1E-10	<i>optional</i>
<b>MaxTimeStep</b>	h	Maximal zulässiger Zeitschritt für die Integration.	positive double ( $10^{-10}$ bis $10^0$ )	1	<i>optional</i>
<b>MinTimeStep</b>	s	Minimal akzeptierter Zeitschritt, bevor der Solver mit einem Fehler abbricht.	positive double ( $10^{-10}$ bis $10^0$ )	1E-12	<i>optional</i>
<b>InitialTimeStep</b>	s	Initiale Zeitschrittgröße (oder konstante Schrittgröße für ExplicitEuler-Integrator).	positive double ( $10^{-10}$ bis $10^0$ )	0.1	<i>optional</i>
<b>NonlinearSolverConvCoeff</b>	---	Koeffizient, der die Konvergenzgrenze des Solvers nichtlinearer Gleichungen reduziert. Wird von Implicit Euler nicht unterstützt.	$0 \leq 1$	0.1	<i>optional</i>
<b>IterativeSolverConvCoeff</b>	---	Koeffizientenreduzierende Konvergenzgrenze des iterativen Gleichungssolvers.	$0 \leq 1$	0.05	<i>optional</i>

Name	Vorgabe Einheit	Beschreibung	Wertebereich	Vor- gab- e	Verwen- dung
DiscMinDx	mm	Minimale Elementbreite für Wanddiskretisierung.	positiv double ( $10^{-10}$ .. $10^0$ )	2	optional
DiscStretchFactor	---	Stretch-Faktor für variable Wanddiskretisierungen:  0 - keine Diskretisierung 1 - Šquidistant > 1 - variabel  siehe <a href="#">spatial discretization algorithm</a> für Details.	positive integer ( $10^0$ .. $10^2$ )	50	optional
(*)ViewfactorTileWidth	m	Maximale Abmessung einer Kachel für die Berechnung der Ansichtsfaktoren.	positive double ( $10^{-10}$ .. $10^0$ )	50	optional
(*)SurfaceDiscretizationDensity	---	Anzahl der OberflŠchendiskretisierungse- lemente einer Wand in jeder Richtung.	$0 \leq 1$	2	optional
(*)ControlTemperatureTolerance	K	Temperaturtoleranz für ideales Heizen oder Kühlen.	positiv double ( $10^{-10}$ .. $10^0$ )	1E-05	optional
(*)KinsolRelTol	---	Relative Toleranz für Kinsol- Solver.	$0 \leq 1$	-	optional
(*)KinsolAbsTol	---	Absolute Toleranz für Kinsol- LŠser.	$0 \leq 1$	-	optional

(\*) - bisher noch nicht verwendet

Ganzzahlige Parameter (siehe Abschnitt [IBK: IntPara](#) für eine Beschreibung des tags [IBK: IntPara](#)):

Name	Beschreibung	Standar- d	Verwen- dung
PreILUWidth	Anzahl der Nicht-Nullen in ILU	---	optional
MaxKrylovDim	Max. GrŠŠe der Krylow-Dimension/max. Anzahl der linearen Iterationen (nur iterative LES)	50	optional
MaxNonlinear	Max. Anzahl der nicht-linearen/Newton-Iterationen	3	optional
MaxOrder	Max. Methodenordnung	5	optional
DiscMaxElementsPerLayer	Max. Anzahl der Diskretisierungselemente pro Materialschicht	20	optional

Name	Beschreibung	Standard	Verwendung
(*) <a href="#">KinsolMaxNonlinear</a>	Max. Iterationen des Kinsol-Solvers	<i>auto</i>	<i>optional</i>

(\*) - bisher noch nicht verwendet

Flags und Optionen (siehe Abschnitt [IBK:Flag](#) für eine Beschreibung des tags [IBK:Flag](#)):

Name	Beschreibung	Standard	Verwendung
(*) <a href="#">DetectMaxTimeStep</a>	Zeitpunkte prüfen, um Mindestabstände zwischen Schritten zu ermitteln und MaxTimeStep anzupassen.	<i>false</i>	<i>optional</i>
(*) <a href="#">KinsolDisableLineSearch</a>	Deaktiviere Liniensuche für stationäre Zyklen.	<i>false</i>	<i>optional</i>
(*) <a href="#">KinsolStrictNewton</a>	Strict Newton für stationäre Zyklen einschalten.	<i>false</i>	<i>optional</i>

(\*) - bisher noch nicht verwendet



Die oben aufgeführten Optionen und Parameter hängen teilweise von den gewählten Zeitintegrationsalgorithmen, LES-Solvern und Vorkonditionierern ab, siehe Tabelle im Abschnitt [Solver-Fähigkeiten](#) unten.

### 16.2.1. Integrator

Der XML-tag [Integrator](#) enthält eine Zeichenkette zur Auswahl eines bestimmten Integrators ([CVODE](#) wird standardmäßig verwendet, wenn das tag fehlt).

Tabelle 22. verfügbare Integratoren

Name	Beschreibung
<a href="#">CVODE</a>	Wählt den CVODE-Integrator aus der Sundials-Bibliothek: implizites Mehrschrittverfahren mit fehlertestbasierter Zeitschrittanpassung und modifiziertem Newton-Raphson für nichtlineare Gleichungssysteme
<a href="#">ExplicitEuler</a>	Expliziter Euler-Integrator (nur zur Fehlersuche, der Parameter <a href="#">InitialTimeStep</a> bestimmt die feste Schrittweite)
<a href="#">ImplicitEuler</a>	Impliziter Euler-Integrator, Einzelschrittlöser mit fehlertestbasierter Zeitschrittanpassung und modifiziertem Newton-Raphson für nichtlineare Gleichungssysteme (nur zur Fehlersuche und für spezielle Tests)

Siehe [Solver-Fähigkeiten](#) für gültige Kombinationen.

16.2.2. Linear equation system (LES) solver

Der XML-tag `LesSolver` enthlt eine Zeichenkette zur Auswahl eines bestimmten Solvers fr die linearen Gleichungssysteme (`KLU` wird standardmSig verwendet, wenn der tag fehlt).

Tabelle 23. verfgbare LES-Solver

Name	Beschreibung
Dense	Direkter dense Solver (nur zur Fehlersuche)
KLU	Direkter Sparse Solver
GMRES	Verallgemeinerte Minimale Residualmethode (iterativer Solver)
BiCGStab	Bikonjugierte stabilisierte Gradientenmethode (iterativer Solver)

Siehe [Solver-Fhigkeiten](#) fr gltige Kombinationen.

16.2.3. Prskonditionierer

Der XML-tag `Preconditioner` enthlt eine Zeichenkette zur Auswahl eines bestimmten Preconditioners, der fr iterative LES-Solver verwendet werden soll (`ILU` wird standardmSig verwendet, wenn das tag fehlt).

Tabelle 24. verfgbare Preconditioners

Name	Beschreibung
ILU	Unvollstndige LU-Faktorisierung (wenn <code>PreILUWidth</code> angegeben ist, wird ILU-T verwendet)

Derzeit sind zwei Varianten des ILU-Preconditioners implementiert. Eine ohne Schwellenwert, bei der die Faktorisierung nur im ursprnglichen Jacobi-Matrixmuster gespeichert wird. Wenn der Benutzer `PreILUWidth` angegeben hat, berechnet die Routine die Faktorisierung und behlt in jeder Zeile die hchsten n-Werte (wobei n durch `PreILUWidth` definiert ist). Diese Methode ist bekannt als *ILU mit Threshold* (ILU-T).

!!

Eine ILU-T-Methode ist nur fr `PreILUWidth` > 3 wirksam. Die minimale Anzahl von Nicht-Nullen in jeder Matrixzeile ist 3, da die Finite-Volumen-Diskretisierung der Wandkonstruktionen bereits ein 3-Diagonal-Muster erzeugt.

16.2.4. Solver-Fhigkeiten

Nicht alle Integratoren und LES-Solver untersttzen alle oben genannten Optionen. Auch knnen nicht alle LES-Solver mit allen Integratoren kombiniert werden. Die folgende Tabelle gibt einen berblick ber die untersttzten Kombinationen und Optionen.

Tabelle 25. Fhigkeiten und untersttzte Flags/Parameter fr die angebotenen Integratoren

Integrator	LES-Solver	Unterstützte Integratorparameter/Flags
CVODE	Dense, KLU, GMRES, BiCGStab	RelTol, AbsTol, MaxTimeStep, MinTimeStep, InitialTimeStep, MaxOrder, NonlinSolverConvCoeff, MaxNonlinIter
Implicit Euler	Dense	RelTol, AbsTol, MaxTimeStep, InitialTimeStep, NonlinSolverConvCoeff, MaxNonlinIter
Explicit Euler	---	InitialTimeStep

Tabelle 26. Fähigkeiten und unterstützte Flags/Parameter für die angebotenen LES-Solver

LES-Solver	Preconditioners	Unterstützte Integratorparameter/Flags
DENSE	---	---
KLU	---	---
GMRES	ILU	PreILUWidth, MaxKrylovDim, IterativeSolverConvCoeff
BiCGStab	ILU	PreILUWidth, MaxKrylovDim, IterativeSolverConvCoeff

## 17. Modellparametrisierung

In diesem Abschnitt werden die verschiedenen Modellparametrisierungsblöcke beschrieben. Modelle werden verwendet, um gleichartige Funktionalität für mehrere Objekte (meist Zonen) zu definieren. Die in den Modellblöcken abgelegten Parameter gelten dann für alle (via Objektlisten) ausgewählten Objekte. Allerdings können objektspezifische Eigenschaften (bspw. Nutzfläche bei Zonen) in die Modell einfließen.

Beispiel 50. Modelldefinitionslisten innerhalb des XML-Elements `Models`

```
<Models>
  <NaturalVentilationModel>
    ...
  </NaturalVentilationModel>

  ... weitere Modell-Definitionsblöcke ...

  <IdealPipeRegisterModel>
    ...
  </IdealPipeRegisterModel>
</Models>
```

### 17.1. Modellüberblick

XML-Tag	Modell/Kurzbeschreibung
Natural VentilationModels	Natürliche Lüftung/Infiltration ! <a href="#">Natürliches Lüftungsmodell</a>
ShadingControlModels	Kontrollmodell für dynamische Verschattung ! <a href="#">Steuerungsmodell für Verschattung</a>
Internal LoadModels	Interne Lasten (Geräte, Personen, Beleuchtung) ! <a href="#">Modell für interne Lasten</a>
Internal MoistureLoadModels	Interne Feuchtelasten (Personen) ! <a href="#">Modell für interne Feuchtelasten</a>
Thermostats	Kontrollmodell für Temperaturregelung ! <a href="#">Modell für Thermostate</a>
(*) HVACControlModels	Kontrollmodell für erweiterte Klimaanlageanlagenregelung ! <a href="#">Anlagensystem-Modell</a>
Ideal HeatingCoolingModels	Ideale Beheizung/Kühlung von Zonen ! <a href="#">Modell für ideale thermische Konditionierung</a>
Ideal SurfaceHeatingCoolingModels	Ideale Fliesenheizsysteme/Fußbodenheizungen/Betonkernaktivierung ! <a href="#">Modell für ideale Fliesenheizungen</a>
Ideal PipeRegisterModels	Idealisierte Rohrregister-Fliesenheizungen ! <a href="#">Modell für idealisierte Rohrregister-Fliesenheizungen</a>
HeatLoadSummationModels	Summationsmodelle für Heizleistungen ! <a href="#">Modell für die Summation von Heiz- und Kühlleistungen</a>
NetworkInterfaceAdapterModels	Adaptermodelle für externe Netzwerkschnittstellen ! <a href="#">Schnittstellen-Modell für die Anbindung externer Anlagennetze</a>

(\*) noch nicht implementiert

## 17.2. Natürliches Lüftungsmodell

Das Modell für natürliche Lüftung definiert den Luftaustausch mit der Außenluft und beinhaltet Nutzerlüftung wie ungewollte Lüftung durch Leckagen (Fugenlüftung/Infiltration). Die natürliche Lüftung wird für bestimmte Zonen aktiviert, indem ein Modellparametersatz **Natural VentilationModel** definiert wird. Über die Objektliste werden die Zonen ausgewählt.



Dieses Modell sollte nicht zusammen mit dem Luftströmungsmodell (Luftströmungs-Netzwerk) verwendet werden.

*Beispiel 51. Drei verschiedene Modellvarianten und deren Parameter beim Modell für natürliche Lüftung/Infiltration*

```
<!-- Lüftungsmodell mit durchgängig konstanter Infiltration/Grundluftwechsel -->
<NaturalVentilationModel id="501" displayName="Zone vent" modelType="Constant">
  <ZoneObjectList>Office zones</ZoneObjectList>
  <IBK:Parameter name="VentilationRate" unit="1/h">0.5</IBK:Parameter>
</NaturalVentilationModel>
```

```

<!-- L ftungsmodell mit zeitabh ngig geregelter Luftwechselrate -->
<NaturalVentilationModel id="502" displayName="Zone vent" modelType="Scheduled">
  <ZoneObjectList>Other zones</ZoneObjectList>
</NaturalVentilationModel>

<!-- L ftungsmodell mit zeitabh ngigem Grundluftwechsel und bedarfsweise erh hter Luftwechselrate -->
<NaturalVentilationModel id="503" displayName="Zone vent" modelType="ScheduledWithBaseACR">
  <ZoneObjectList>Special zones</ZoneObjectList>
  <!-- Komfortbereich wird durch min und max Temperaturen angegeben :
  Erh hte L ftungsrate verwenden, wenn Raumtemperatur 24 C  berschreitet und es drau en k lter ist. -->
  <IBK:Parameter name="MaxAirTemperature" unit=" C">24</IBK:Parameter>
  <!-- Erh hung der L ftungsrate verwenden, wenn Raumtemperatur 18 C unterschreitet und es drau en w rmer ist.
  -->
  <IBK:Parameter name="MinAirTemperature" unit=" C">18</IBK:Parameter>
  <!--  ber 10 m/s wird die erh hte L ftung ausgeschaltet. -->
  <IBK:Parameter name="MaxWindSpeed" unit="m/s">10</IBK:Parameter>
</NaturalVentilationModel>

```



Es darf nur ein L ftungsmodellblock pro Zone gelten. Es d rfen also nicht zwei L ftungsmodellbl cke mit Objektlisten definiert werden, die beide die gleiche(n) Zone(n) enthalten.

Das **NaturalVentilationModel** unterst tzt folgende XML-Attribute:

*Tabelle 27. Attribute*

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Kennung des nat�rlichen L�ftungsmodells	���	<i>erforderlich</i>
<b>displayName</b>	Anzeigename/Beschreibung	string	<i>optional</i>

Attribut	Beschreibung	Format	Verwendung
<b>model Type</b>	<p>Setzt den Typ des Lüftungsmodells</p> <p>¥ <b>Constant</b> - Konstante Luftwechselrate</p> <p>¥ <b>Scheduled</b> - Luftwechselrate ändert sich nach einem definierten Zeitplan. Dieses Modell sollte verwendet werden, wenn ein spezielles Lüftungsmodell über die FMU-Schnittstelle dazugeschaltet wird.</p> <p>¥ <b>ScheduledWithBaseACR</b> - Es werden ein Grundluftwechsel und eine <i>Erhöhung</i> des Luftwechsels jeweils als zeitabhängige Zeitpläne gegeben. Die Aktivierung des zusätzlichen Luftwechsels hängt an zusätzlichen Kriterien. Speziell für diesen Modelltyp sind die Grenztemperaturen <i>MaxAirTemperature</i> und <i>MinAirTemperature</i> als konstante Parameter gegeben. Dies ist eine spezielle Ausprägung des allgemeineren Falles <i>ScheduledWithBaseACRDynamicTLimit</i> und primär zum Testen vorgesehen.</p> <p>¥ <b>ScheduledWithBaseACRDynamicTLimit</b> - Es werden ein Grundluftwechsel und eine <i>Erhöhung</i> des Luftwechsels jeweils als zeitabhängige Zeitpläne gegeben. Die Aktivierung des zusätzlichen Luftwechsels hängt an zusätzlichen Kriterien. Dazu zählen die Unter- und Überschreitung der Grenztemperaturen <i>MaxAirTemperature</i> und <i>MinAirTemperature</i>, welche als zeitabhängige Zeitpläne gegeben sind.</p> <p>¥ (*)<b>Realistic</b> - Lüftungsrate ergibt sich aus physikalischen Bedingungen (Wind, Temperaturdifferenzen, Fensteröffnungszustand, etc.)</p>	key	<i>erforderlich</i>

(\*) wird noch nicht verwendet/noch nicht implementiert

Gleitkommazahlen-Parameter (siehe [IBK:Parameter](#) für eine Beschreibung des Elementtyps [IBK:Parameter](#)):

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
<b>VentilationRate</b>	1/h	Konstante Luftwechselrate	$\hat{V} \in ]0,0]$	<i>erforderlich für Modelltyp <b>p Constant</b></i>



Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
VentilationMaxAirTemperature	C	obere Raumluft-Grenztemperatur	$\hat{E} = \hat{E} 100.0 \hat{E}$	erforderlich für Modelltyp <i>p</i> ScheduleWithBaseACR
VentilationMinAirTemperature	C	untere Raumluft-Grenztemperatur	$\hat{E} = \hat{E} 100.0 \hat{E}$	erforderlich für Modelltyp <i>p</i> ScheduleWithBaseACR
MaxWindSpeed	m/s	maximale Windgeschwindigkeit für erhöhte Lüftung	$\hat{E} = \hat{E} 0.0 \hat{E}$	erforderlich für Modelltyp <i>p</i> ScheduleWithBaseACR und ScheduleWithBaseACRDynamicTLimit

Beim Modelltyp *Constant* wird durchweg eine konstante Luftwechselrate (Parameter *VentilationRate*) verwendet.

Beim Modelltyp *Scheduled* wird die Luftwechselrate aus einem Zeitplan (Parameter *VentilationRateSchedule*, siehe [Zeitplan](#)) entnommen. Weitere Parameter sind nicht notwendig.

Nachfolgend finden Sie ein Beispiel für einen Zeitplan, der den Parameter *VentilationRateSchedule* für ein solches Modell der geplanten natürlichen Lüftung bereitstellt:

*Beispiel 52. Zeitplan, der den Parameter VentilationRateSchedule bereitstellt*

```
<ScheduleGroup objectList="All zones">
  <!-- jeden Tag zwischen 6-10 -->
  <Schedule type="AllDays">
    <DailyCycles>
      <DailyCycle interpolation="Constant">
        <TimePoints>0 6 10</TimePoints>
        <Values>VentilationRateSchedule [1/h]:0 0.4 0</Values>
      </DailyCycle>
    </DailyCycles>
  </Schedule>
</ScheduleGroup>
```

```

</Schedule>
<!-- Dienstag keine Lüftung -->
<Schedule type="Tuesday">
  <DailyCycles>
    <DailyCycle interpolation="Constant">
      <TimePoints>0</TimePoints>
      <Values>VentilationRateSchedule [1/h]:0</Values>
    </DailyCycle>
  </DailyCycles>
</Schedule>
<!-- Wochenende nur am Nachmittag -->
<Schedule type="WeekEnd">
  <DailyCycles>
    <DailyCycle interpolation="Constant">
      <TimePoints>0 14 16</TimePoints>
      <Values>VentilationRateSchedule [1/h]:0 0.1 0</Values>
    </DailyCycle>
  </DailyCycles>
</Schedule>
</ScheduleGroup>

```

Bei den Modelltypen *ScheduledWithBaseACR* und *ScheduledWithBaseACRDynamicTLimit* wird ein Grundluftwechsel (Schedule *VentilationRateSchedule*) analog zum Modelltyp *Scheduled* verwendet. Unter bestimmten Bedingungen wird dieser Grundluftwechsel durch ein zusätzlichen Luftwechsel entsprechend eines gegebenen Zeitplans in *VentilationRateIncreaseSchedule* erhöht.

Die Luftwechselrate wird berechnet:

```

n = n_Grundluftwechsel // wenn Bedingungen nicht erfüllt
n = n_Grundluftwechsel + n_erhöhterLuftwechsel // wenn Bedingungen erfüllt

```



Bei Definition der *VentilationRateIncreaseSchedule* ist zu beachten, dass dies die zusätzliche Lüftungsrate zum Grundluftwechsel ist.

**Beispiel 53.** Zeitplan für Modelltyp *ScheduledWithBaseACR*; Grundluftwechsel = 0.5 1/h, Luftwechsel bei Erhöhung = 2 1/h = (0.5 + 1.5) 1/h

```

<ScheduleGroup objectList="All zones">
  <Schedule type="All Days">
    <DailyCycles>
      <DailyCycle interpolation="Constant">
        <TimePoints>0</TimePoints>
        <!-- Basis Luftwechsel rate -->
        <Values>
          VentilationRateSchedule [1/h]:0.5;
          VentilationRateIncreaseSchedule [1/h]:1.5
        </Values>
      </DailyCycle>
    </DailyCycles>
  </Schedule>
</ScheduleGroup>

```

Der Modelltyp *ScheduledWithBaseACRDynamicTLimit* verlangt zusätzlich den *ScheduleVentilationMaxAirTemperatureSchedule* und *VentilationMinAirTemperatureSchedule*. Die dazugehörigen Grenztemperaturen werden dann dynamisch ausgewertet. Konstante Temperaturannahmen werden hiervon umfasst und können durch passende Zeitpläne generiert werden:

*Beispiel 54. Zeitplan für Modelltyp ScheduledWithBaseACRDynamicTLimit; konstante Grenztemperaturen (Minimum: 23 C, Maximum 100 C)*

```
<ScheduleGroup objectList="All zones">
  <Schedule type="AllDays">
    <DailyCycles>
      <DailyCycle interpolation="Constant">
        <TimePoints>0</TimePoints>
        <!-- Konstante Grenztemperaturen -->
        <Values>
          VentilationMaxAirTemperatureSchedule [C]: 100;
          VentilationMinAirTemperatureSchedule [C]: 23
        </Values>
      </DailyCycle>
    </DailyCycles>
  </Schedule>
</ScheduleGroup>
```

### 17.2.1. Regelbedingungen

Die Komfortlüftung (Modelltyp *ScheduledWithBaseACR*) folgt einer einfachen Logik:

- ⌘ der Grundluftwechsel reicht aus, solange die Raumlufttemperatur zwischen den gegebenen Parametern *MinAirTemperature* und *MaxAirTemperature* liegt
- ⌘ sobald der Komfortbereich verlassen wird, wird die zusätzliche Luftwechselrate angewendet, \_ aber nur, falls eine Lüftung hilfreich ist\_. D.h. durch die erhöhte Lüftung muss die Raumlufttemperatur in Richtung Komfortzone bewegt werden.

Beispiel Kühlung im Sommerfall

- ⌘ *MaxAirTemperature* = 26 C
- ⌘ Falls die Raumlufttemperatur > 26 C wird, kann die erhöhte Lüftung benutzt werden, aber nur solange die Außenlufttemperatur kleiner als die aktuelle Raumlufttemperatur ist.

### 17.2.2. Ausgabegrößen

Das Lüftungsmodell generiert folgende vektorwertige Ergebnisgrößen:

- ⌘ *VentilationHeatFlux* in [W]
- ⌘ *VentilationRate* in [1/h] und



Da es mehrere Lüftungsmodellinstanzen geben kann, sucht das jeweilige Zonenmodell zunächst, welche Modellinstanz eine Ergebnisgröße liefert und erstellt dann die

## 17.3. Steuerungsmodell für Verschattung

Ein Verschattungsregelungsmodell ist eine spezielle Art von Regelungsmodell, das einen Signalwert zwischen 0 (keine Verschattung) und 1 (volle Verschattung) zurückgibt. Das tatsächliche Ausmaß der Verschattung bzw. die Reduzierung der solaren Gewinne wird durch den Verschattungs-Parameterblock (**Shading**, siehe [Fensterverschattung](#)) bestimmt. Somit kann das gleiche Regelmodell für verschiedene Verschattungseinrichtungen verwendet werden. Da es bei Verschattungseinrichtungen keinen expliziten Zonenbezug gibt, werden Verschattungskontrollmodelle über ihre eindeutige ID referenziert.

### Beispiel 55. Parameterdefinition für Verschattungsregelungsmodell

```
<Model s>
  <ShadingControlModel s>
    <!-- ShadingControlModel liefert einen Wert zwischen 0 und 1
    0 = keine Reduktion (Verschattung offen)
    1 = volle Reduktion (Verschattung geschlossen)
    -->
    <ShadingControlModel id="2000" displayName="Global horizontal sensor controller" sensorId="50000">
      <IBK:Parameter name="MaxIntensity" unit="W/m2">300</IBK:Parameter>
      <IBK:Parameter name="MinIntensity" unit="W/m2">150</IBK:Parameter>
    </ShadingControlModel>
  </ShadingControlModel s>
</Model s>
```

Das **ShadingControlModel** unterstützt folgende XML-Attribute:

Tabelle 28. Attribute

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Kennung des Modells	int	erforderlich
<b>displayName</b>	Anzeigename/Beschreibung	string	optional

Das Verschattungskontrollmodell verlangt zwei Parameter **MaxIntensity** und **MinIntensity** und implementiert eine digitale Regelung mit Hysterese. Zunächst muss die Globalstrahlungsintensität auf den Sensor den oberen Grenzwert (**MaxIntensity**) überschreiten, wonach die Verschattung geschlossen wird (Kontrollmodell liefert 1). Danach muss die Strahlungsintensität zunächst unter die untere Grenze (**MinIntensity**), bevor die Verschattung wieder geöffnet wird (Kontrollmodell liefert 0).

Für die Auswertung wird eine Horizontalstrahlung benötigt. Dafür muss eine Oberfläche ausgewählt werden und als **sensorId** angegeben werden.

Möglich sind hier 3 Optionen:

- allgemeiner Sensor auf einer Fläche (siehe [Zusätzliche Strahlungssensoren](#))

- ¥ ID eines Fensters (eigentlich ID des *embedded object*, welches das Fenster enthält); hier wird die Globalstrahlung durch das Fenster als Eingangsgröße verwendet, einschließlich eventueller externer Verschattung bzw. Eigenverschattung
- ¥ ID einer opaquen Fläche; hier wird die Globalstrahlung auf eine opaque Fläche als Eingangsgröße verwendet, einschließlich eventueller externer Verschattung bzw. Eigenverschattung

Damit diese IDs eindeutig auflösbar sind, müssen Sensoren, Fenster und Konstruktionen global eindeutige IDs tragen (siehe auch [Eindeutigkeitsforderungen für Definitions-IDs](#)).

### 17.3.1. Ausgabegrößen

Das Verschattungssteuerungsmodell liefert als Ergebnisgrößen:

- ¥ **ShadingControlValue** - Steuerungssignal für Verschattung: 0 - komplett offen, 1 - komplett geschlossen, Zwischenwerte sind möglich
- ¥ **SolarIntensityOnShadingSensor** - Solarstrahlungsintensität in [W/m<sup>2</sup>] auf ausgewählten Sensor, der für die Regelung verwendet wird

## 17.4. Modell für interne Lasten

Das Modell für interne Lasten wird verwendet, um die Wärmelasten von Geräten, Personen und Beleuchtung für Zonen zu definieren. Interne Lasten werden genauso definiert wie natürliche Lüftungsmodelle. Der Objektlisten-tag **ZoneObjectList** identifiziert die Zonen, in denen interne Lasten berücksichtigt werden sollen. Wie auch beim Modell für natürliche Lüften dürfen Zonen immer nur einmal referenziert werden (es dürfen nicht zwei interne Lastmodelle existieren, die sich auf dieselben Zonen beziehen).

*Beispiel 56. Definitionsblock für interne Lasten*

```
<InternalLoadsModel id="200" modelType="Scheduled">
  <ZoneObjectList>Office zones</ZoneObjectList>
  <IBK:Parameter name="RadiantFraction" unit="---">0.5</IBK:Parameter>
</InternalLoadsModel>
```

Das **InternalLoadsModel** unterstützt folgende XML-Attribute:

*Tabelle 29. Attribute*

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Kennung des Modells	ℵ-ℵ	<i>erforderlich</i>
<b>displayName</b>	Anzeigename/Beschreibung	string	<i>optional</i>

Attribut	Beschreibung	Format	Verwendung
model Type	<p>Gibt an, wie die internen Lasten angesetzt werden sollen</p> <p>¥ <b>Constant</b> - Konstante GerŠte-, Personen- und Beleuchtungsenergielasten</p> <p>¥ <b>Scheduled</b> - Lasten werden Ÿber Zeitplanparameter bereitgestellt.</p>	key	erforderlich

FlieŠkommaparameter (siehe [IBK:Parameter](#) fŸr eine Beschreibung des Elementtyps [IBK:Parameter](#)):

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
EquipmentHeatLoadPerArea	W/m <sup>2</sup>	Komplette GerŠtebelastung pro ZonennutzflŠche	$\hat{E} \geq 0.0 \hat{E}$	erforderlich fŸr Konstantes Modell
PersonHeatLoadPerArea	W/m <sup>2</sup>	Komplette PersonenwŠrmelast pro ZonennutzflŠche	$\hat{E} \geq 0.0 \hat{E}$	erforderlich fŸr Konstantes Modell
LightingHeatLoadPerArea	W/m <sup>2</sup>	Komplette WŠrmelast aus Beleuchtung pro ZonennutzflŠche	$\hat{E} \geq 0.0 \hat{E}$	erforderlich fŸr Konstantes Modell
EquipmentRadiationFraction	---	Prozentualer Anteil der WŠrme der GerŠte, der durch Strahlung emittiert wird	$\hat{E} \geq 0.0 \hat{E}$	erforderlich
PersonRadiationFraction	---	Prozentualer Anteil der WŠrme der Personen, der durch Strahlung emittiert wird	$\hat{E} \geq 0.0 \hat{E}$	erforderlich
LightingRadiationFraction	---	Prozentualer Anteil der WŠrme der Beleuchtung, der durch Strahlung emittiert wird	$\hat{E} \geq 0.0 \hat{E}$	erforderlich



Die *ZonennutzflŠche* ist nicht zwingend die *GrundflŠche* einer Zone sondern wird aus dem Parameter *Area* der Zonendefinition gewŠhlt. Dadurch ist es mŠglich, z.B. im Dachgeschoss mit SchrŠgen, die tatsŠchlich nutzbare FlŠche zu definieren und zu verwenden. Deshalb wird der *Area* Parameter in allen Zonen benŠtigt, fŸr die ein [Internal LoadsModel](#) angewendet werden soll.

Die Parameter *xxxRadiationFraction* geben an, welcher Prozentsatz der berechneten internen Lasten als Strahlungsfluss flŠchengewichtet auf opake OberflŠchen, die die Zone umschlieŠen, aufgebracht werden

soll.

Der Modelltyp **Constant** übernimmt die internen Lasten aus den Parametern (siehe oben).

Wenn der Modelltyp **Scheduled** verwendet wird, werden die tatsächlichen Lasten aus dem Zeitplan entnommen.

Die folgenden Zeitplanparameter sind erforderlich:

¥ **EquipmentHeatLoadPerAreaSchedule** [W/m<sup>2</sup>]

¥ **PersonHeatLoadPerAreaSchedule** [W/m<sup>2</sup>]

¥ **LightingHeatLoadPerAreaSchedule** [W/m<sup>2</sup>]

### 17.4.1. Ausgaben

Das Modell stellt folgende Ausgangsgrößen zur Verfügung:

¥ **ConvectiveEquipmentHeatLoad** [W]

¥ **ConvectivePersonHeatLoad** [W]

¥ **ConvectiveLightingHeatLoad** [W]

¥ **RadiantEquipmentHeatLoad** [W]

¥ **RadiantPersonHeatLoad** [W]

¥ **RadiantLightingHeatLoad** [W]

Dies sind vektoriell dargestellte Größen, die in Ausgangsdefinitionen referenziert werden müssen, z. B. mit: **ConvectiveEquipmentHeatLoad[id=3]** für die konvektive Gerüstlast in Zone #3.

## 17.5. Modell für interne Feuchtelasten

Das Modell beschreibt analog zum *Internal loads model* die inneren Feuchtelasten im Raum. Dabei findet die Feuchteproduktion durch Personen Berücksichtigung. Erweiterungen sind geplant. Der Objektlistentag **ZoneObjectList** identifiziert die Zonen, in denen Feuchtelasten berücksichtigt werden sollen. Alle Zonen dürfen durch maximal eine Feuchtelastenmodell referenziert werden.

*Beispiel 57. Definitionsblock für interne Feuchtelasten*

```
<InternalMoistureLoadsModel id="300" modelType="Scheduled">
  <ZoneObjectList>Office zones</ZoneObjectList>
</InternalMoistureLoadsModel>
```

Das **InternalMoistureLoadsModel** unterstützt folgende XML-Attribute:

*Tabelle 30. Attribute*

Attribut	Beschreibung	Format	Verwendung
<code>id</code>	Kennung des Modells	<code>&gt;0</code>	<i>erforderlich</i>
<code>displayName</code>	Anzeigename/Beschreibung	string	<i>optional</i>
<code>modelType</code>	Gibt an, wie die internen Lasten angesetzt werden sollen  $\text{Constant}$ - Konstante Personenfeuchtelasten $\text{Scheduled}$ - Lasten werden über Zeitplanparameter bereitgestellt.	key	<i>erforderlich</i>

Fließkommaparameter (siehe [IBK:Parameter](#) für eine Beschreibung des Elementtyps [IBK:Parameter](#)):

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
<code>PersonHeatMoistureLoadPerArea</code>	kg/m <sup>2</sup> s	Personenfeuchteproduktion pro Zonnennutzfläche	<code>&gt;=0.0</code>	<i>erforderlich für Konstantes Modell</i>

Der Modelltyp `Constant` übernimmt die internen Feuchtelasten aus den Parametern. Der Modelltyp `Scheduled` verwendet die Lasten aus dem Zeitplan:

$\text{PersonHeatMoistureLoadPerAreaSchedule}$  [kg/m<sup>2</sup>s]

Die Existenz dieses Zeitplanparameters ist zwingend erforderlich.

### 17.5.1. Ausgaben

Das Modell stellt folgende Ausgangsgrößen zur Verfügung:

$\text{PersonMoistureLoad}$  [W]

$\text{PersonMoistureEnthalpyFlux}$  [W]

Die Ausgaben sind vektoriell und müssen mit Zonenspezifikation referenziert werden, z. B. mit: `PersonMoistureEnthalpyFlux[id=3]` für die Enthalpieströme in Verbindung mit Personenfeuchteproduktion in Zone #3.

## 17.6. Modell für Thermostate

Das Thermostatmodell beschreibt, auf welche Raumsollwerte konditioniert werden soll. Angegeben werden können Heiz- und/oder KÜhlsolltemperaturen für die Raumluft oder operative Raumluft. Der Objektlisten-tag `ZoneObjectList` identifiziert die Zonen, in denen Thermostate berücksichtigt werden sollen. Wie auch beim Lüftungsmodell darf nur ein Modell pro Zone existieren.



## Beispiel 58. Definitionsblock für Thermostate

```
<!-- A thermostat with constant heating and cooling set points. Uses air temperature as sensor value. -->
<Thermostat id="1001" displayName="Constant air temperature thermostat" modelType="Constant">
  <ZoneObjectList>All zones</ZoneObjectList>
  <!-- Heating starts below 22 C -->
  <IBK:Parameter name="HeatingSetpoint" unit="C">22</IBK:Parameter>
  <!-- Cooling starts above 26 C -->
  <IBK:Parameter name="CoolingSetpoint" unit="C">26</IBK:Parameter>
  <!-- P-controller is accurate to 0.2 K -->
  <IBK:Parameter name="TemperatureTolerance" unit="K">0.2</IBK:Parameter>
  <!-- Control temperature is "Air temperature", this is the default and could be omitted -->
  <TemperatureType>AirTemperature</TemperatureType>
  <!-- Controller type Analog is the default, so we could omit this-->
  <ControllerType>Analog</ControllerType>
</Thermostat>
```

Das **Thermostat**-Element unterstützt folgende XML-Attribute:

Tabelle 31. Attribute

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Kennung des Modells	int	erforderlich
<b>displayName</b>	Anzeigename/Beschreibung	string	optional
<b>modelType</b>	Gibt an, wie die Thermostat-Parameter angesetzt werden sollen  • <b>Constant</b> - Konstante Sollwerte • <b>Scheduled</b> - Sollwerte werden über Zeitplanparameter bereitgestellt.	key	erforderlich

Fließkommaparameter (siehe **IBK:Parameter** für eine Beschreibung des Elementtyps **IBK:Parameter**):

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
<b>HeatingSetpoint</b>	C	konstanter Heizsollwert	$< \text{CoolingSetpoint}$	erforderlich für Modelltyp <b>Constant</b>
<b>CoolingSetpoint</b>	C	konstanter KÜhlsollwert	$> \text{HeatingSetpoint}$	erforderlich für Modelltyp <b>Constant</b>

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
TemperatureTolerance	K	Toleranz für analoges Thermostat und Normalisierungswert	> 0	erforderlich für Controlle rtyp Analog
TemperatureBand	K	Totband für digitales Thermostat	> 0	erforderlich für Controlle rtyp Digital

Der Modelltyp **Constant** übernimmt die Sollwerte aus den Parametern (siehe oben).

Wenn der Modelltyp **Scheduled** verwendet wird, werden die tatsächlichen Sollwerte aus dem Zeitplan entnommen. Dafür sind folgende Zeitplanparameter erforderlich:

¥ HeatingSetpointSchedule [C]

¥ CoolingSetpointSchedule [C]



Ein Thermostat hält nur die Sollwerte für die Zone. Eine Konditionierung der Zone erfolgt erst, wenn zusätzlich eine Heizungs- und/oder Kühlmodell für die Zone integriert ist. Bei den Zeitplänen ist immer darauf zu achten, dass der Heizsollwert < KÜhlsollwert ist.

### 17.6.1. TemperatureType

Das XML-Element **TemperatureType** enthält eine Zeichenkette zur Auswahl eines bestimmten Typs (**AirTemperature** wird standardmäßig verwendet, wenn das Element fehlt).

Tabelle 32. Verfügbare Temperatursensoren

Name	Beschreibung
AirTemperature	Als Referenztemperatur wird die Raumlufthtemperatur verwendet.
OperativeTemperature	Als Referenztemperatur wird die operative Raumlufthtemperatur verwendet. Diese setzt sich aus der mittleren Oberflächentemperatur aller Innenoberflächen und aus der Raumlufthtemperatur zusammen. Die Anteile betragen jeweils 50%.

### 17.6.2. ControllerType

Das XML-Element **ControllerType** enthält eine Zeichenkette zur Auswahl des gewünschten Signaltyps (**Analog** wird standardmäßig verwendet, wenn das Element fehlt).

Tabelle 33. Verfügbare Schalt-/Regelsignale

Name	Beschreibung
Analog	Die Abweichung zwischen Solltemperatur und Sensortemperatur wird durch die <code>TemperatureTolerance</code> geteilt zurückgeliefert. Je weiter von 0 verschieden das Sensorsignal ist, umso größer ist die Abweichung. Ein Wert < 1 besagt, dass die Abweichung noch in der Toleranz liegt.
Digital	Ein digitaler Controller mit Hysterese wird verwendet.

Beim Typ `Analog` muss der Parameter `TemperatureTolerance` angegeben werden, welche die nominal erlaubte Regelabweichung definiert. Wird z.B. die Heizsolltemperatur um genau diese Toleranz unterschritten, so liefert der Regler eine 1 zurück (bei größeren Abweichungen entsprechend höhere Werte).

Beim Typ `Digital` muss der Parameter `TemperatureBand` angegeben werden. Der Regler regelt dann im Bereich;

obere Grenze = Sollwert + `TemperatureBand`  
untere Grenze = Sollwert - `TemperatureBand`

### 17.6.3. Ausgaben

Das Modell liefert vektorwertige Modellergebnisgrößen, wobei der Vektorindex die jeweilige Zonen-ID ist.

¥ `HeatingControlValue [---]` - Steuersignal für die Heizungsanlage: 0 - aus, 1 - maximal an, Wertebereich unbegrenzt

¥ `CoolingControlValue [---]` - Steuersignal für die Klimaanlage: 0 - aus, 1 - maximal an, Wertebereich unbegrenzt

¥ `ThermostatHeatingSetpoint [C]` - Setpoint, der für die Heizung verwendet wurde

¥ `ThermostatCoolingSetpoint [C]` - Setpoint, der für die Kühlung verwendet wurde



Es kann mehrere Thermostat-Modell-Instanzen im Gebäude geben. Da die Ergebnisgrößen von der jeweiligen Modellinstanz selbst zur Verfügung gestellt werden, muss beim Zugriff auf die jeweiligen zonenspezifischen Regelgrößen (`XXXControlValue`) das richtige Modell adressiert werden. In der Praxis kann das so geschehen, dass ein nachfolgendes Modell einfach optionale Eingangsreferenzen für Regelgrößen einer Zonen an *alle* Thermostatmodelle schickt. Es darf dann nur exakt ein Modell eine Ergebnisgröße liefern, die dann verwendet wird.

## 17.7. Anlagensystem-Modell

!NOCH NICHT IMPLEMENTIERT!

Ein Anlagensystem-Modell wandelt Regelinformationen aus einem Thermostat und ggfs. anderen Prozessbedingungen in Regelgrößen für spezifische Heizsysteme um. So können auch Priorisierungen

implementiert werden.

Ein Anlagensystem-Modell ist optional - ohne ein solches System kann ein Heizungssystem auch direkt die Kontrollgrößen eines Thermostats abgreifen.

Beispiel 59. Anlagensystemmodell für die Umsetzung einer idealen Heizung

```
<HVACControlSystem id="200" modelType="Heating">
  <HeatingSystems>Ideal</HeatingSystem>
  <Priority>Parallel</Priority>
</HVACControlSystem>
```

TODO :

17.8. Modell für ideale thermische Konditionierung

Das Modell beschreibt eine ideale thermisches Raumluftkonditionierung. Der Objektlisten-tag `ZoneObjectList` identifiziert die Zonen, in denen das Modell berücksichtigt werden sollen. Es darf nur ein Modell pro Zone existieren (d.h. bei mehreren `IdealHeatingCoolingModel` Definitionen dürfen die Objektlisten nicht gleiche Zonen enthalten).

Beispiel 60. Definitionsblock für ideale thermische Konditionierung

```
<IdealHeatingCoolingModel id="200" displayName="Air heating">
  <ZoneObjectList>Office zones</ZoneObjectList>
  <IBK:Parameter name="MaxHeatingPowerPerArea" unit="W/m2">50</IBK:Parameter>
  <IBK:Parameter name="MaxCoolingPowerPerArea" unit="W/m2">20</IBK:Parameter>
</IdealHeatingCoolingModel>
```

Das `IdealHeatingCoolingModel` unterstützt folgende XML-Attribute:

Tabelle 34. Attribute

Attribut	Beschreibung	Format	Verwendung
<code>id</code>	Kennung des Modells	Int	erforderlich
<code>displayName</code>	Anzeigename/Beschreibung	string	optional

Fließkommaparameter (siehe `IBK:Parameter` für eine Beschreibung des Elementtyps `IBK:Parameter`):

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
MaxHeatingPowerPerArea	W/m <sup>2</sup>	maximale flächenbezogene Heizleistung in Bezug auf Raumnutzfläche	$\hat{P} \in [0.0]$	erforderlich
MaxCoolingPowerPerArea	W/m <sup>2</sup>	maximale flächenbezogene Kälheleistung in Bezug auf Raumnutzfläche	$\hat{P} \in [0.0]$	erforderlich
Kp	---	Faktor für den proportionalen Teil des Controllers	$\hat{P} \in [0.0]$	erforderlich
Ki	---	Faktor für den integralen Teil des Controllers	$\hat{P} \in [0.0]$	optional

Damit das Modell auf die jeweilige Zone angewendet wird, ist zwingend das **Thermostat** nötig, welches für die gleichen Zone parametrisiert sein muss. Dieses liefert dann eine Ergebnisgröße `Model (<thermostat_id>).HeatingControlValue[<zone id>]`.

■

Ein zwischengeschaltetes **HVACControlSystem** kann optional verwendet werden (auch als FMU), welches ein konkretisiertes Regelsignal `Model (<HVACControlSystem_id>).IdealHeatingControlValue[<zone id>]` liefert. Das **IdealHeatingCoolingModel**-Modell sucht *zuerst* nach einer gültigen Variable vom HVACControlSystem-Modell, und dann nach dem Thermostat.

!DIESES IST NOCH NICHT IMPLEMENTIERT!

### 17.8.1. Heiz- und Kälheleistung

Das Regelsignal **HeatingControlValue** bzw. **IdealHeatingControlValue** wird als Wert zwischen 0..1 interpretiert. Werte außerhalb dieser Grenzen werden abgeschnitten. Bei 1 läuft die Heizung mit maximaler Heizlast. Bei 0 ist die Heizung aus. Dazwischen wird linear interpoliert.

Die Kälhlung ist analog definiert. Bei einem Kontrollwert von 1 läuft die Kälhlung mit maximaler Kraft, bei 0 ist sie aus.

Das so bestimmte Regeleingangssignal geht in einen P-Regler ein. Falls der **Ki** Parameter gegeben ist, wird stattdessen ein PI-Regler verwendet. Die so berechnete Heiz- und Kälheleistung wird durch den **MaxHeatingPowerPerArea**-Parameter bzw. **MaxCoolingPowerPerArea**-Parameter begrenzt.

### 17.8.2. Ausgaben

Ergebnisgrößen des Modells sind:

¥ **IdealHeatingLoad** [W]

¥ **IdealCoolingLoad** [W]

Analog zu Lüftungswärmeverlusten werden diese zonenspezifischen Ausgangsgrößen als vektorwertige Ergebnisgrößen bereitgestellt. Z. B. ist `Model<IdealHeatingCoolingModel_id>.IdealHeatingLoad[id=3]` die Heizlast in Zone #3.



Die Kältebelastung ist positiv definiert, geht jedoch als negative Flussgröße in die Raumenergiebilanz ein. In Ausgaben wird die Kältebelastung `IdealCoolingLoad` jedoch immer als positive Größe ausgegeben.

## 17.9. Modell für ideale Fliesenheizungen

Das Modell beschreibt ein ideales thermisches Konditionierungsmodell für eine Fliesenheizung. Dies kann eine Fußbodenheizung sein, eine Kapillarrohrmatte, ein elektrisches Heizregister oder eine Betonkernaktivierung. Der Wärmeübertragungsmechanismus ist nicht abgebildet, es wird lediglich Wärme der beheizten/gekühlten Konstruktionsschicht zugefügt oder entfernt.

*Beispiel 61. Definitionsblock für ideale Fliesenheizungen*

```
<IdealSurfaceHeatingCoolingModel id="701" displayName="Floor heating">
  <!-- Use thermostat in zone 1 for control -->
  <ThermostatZoneId>1</ThermostatZoneId>
  <ConstructionObjectList>Floor</ConstructionObjectList>
  <!-- Maximum heating power per construction/surface area, here: 10 m2 * 150 W/m2 = 1500 W -->
  <IBK:Parameter name="MaxHeatingPowerPerArea" unit="W/m2">150</IBK:Parameter>
</IdealSurfaceHeatingCoolingModel>
```

Das `IdealSurfaceHeatingCoolingModel` unterstützt folgende XML-Attribute:

*Tabelle 35. Attribute*

Attribut	Beschreibung	Format	Verwendung
<code>id</code>	Kennung des Modells	<code>int</code>	<i>erforderlich</i>
<code>displayName</code>	Anzeigename/Beschreibung	string	<i>optional</i>

Fließkommaparameter (siehe `IBK:Parameter` für eine Beschreibung des Elementtyps `IBK:Parameter`):

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
<code>MaxHeatingPowerPerArea</code>	W/m2	maximale flächenbezogene Heizleistung in Bezug auf die Fläche der beheizten Konstruktion	<code>&gt;=0.0</code>	<i>erforderlich</i>
<code>MaxCoolingPowerPerArea</code>	W/m2	maximale flächenbezogene Kälteleistung in Bezug auf die Fläche der beheizten Konstruktion	<code>&gt;=0.0</code>	<i>erforderlich</i>

Ein `IdealSurfaceHeatingCoolingModel` kann die Heiz- und K hlleistung in mehreren Konstruktionen berechnen. Daf r identifiziert der Objektlisten-tag `ConstructionObjectList` die Konstruktion, welche durch das Modell beheizt werden. Jede beheizte Konstruktion darf nur von einem Modell angesprochen werden.



Die in der Objektliste referenzierten Konstruktionen m ssen einen Konstruktionstyp haben, in dem eine aktive Schicht definiert ist (siehe [Aktive/beheizte Schichten](#)).

Die Fl schenheizung/-k hlung wird durch ein Thermostat gesteuert. Die Zone, in der sich das Thermostat befindet, wird im Element `ThermostatZoneId` angegeben.

### 17.9.1. Heiz- und K hlleistung

Das Regelsignal `HeatingControlValue` wird als Wert zwischen 0..1 interpretiert. Werte au erhalb dieser Grenzen werden abgeschnitten. Bei 1 l uft die Heizung mit maximaler Heizlast, entsprechend des `MaxHeatingPowerPerArea`-Parameters. Bei 0 ist die Heizung aus. Dazwischen wird linear interpoliert.

Die K hlung ist genauso definiert. Bei einem Kontrollwert von 1 l uft die K hlung mit maximaler Kraft, bei 0 ist sie aus.

### 17.9.2. Ausgaben

Ergebnisgr ssen des Modells sind:

¥ `ActiveLayerThermalLoad [W]`

Diese konstruktionsspezifischen Gr ssen werden als vektorwertige Ergebnisgr ssen bereitgestellt, mit Zugriff  ber die ID der Konstruktionsinstanz: Z. B. ist `Model<IdealSurfaceHeatingCoolingModel_id>.ActiveLayerThermalLoad[id=3]` die Heizlast in Konstruktion #3.



Die `ActiveLayerThermalLoad` ist positiv bei Beheizung der Fl sche und negativ bei K hlung. Die W rme-/K hlleistung ist eine absolute Gr  e (nicht fl schenbezogen).

## 17.10. Modell f r idealisierte Rohrregister-Fl schenheizungen

Das Modell beschreibt ein Rohrregister, welches zur Beheizung/K hlung von Fl schen verwendet werden kann. Dabei wird das Rohrnetzwerk im Geb ude nicht abgebildet, sondern eine Vorlauftemperatur als gegeben angenommen (daher *idealisiert*). Das Rohrregistermodell ist auch deshalb *ideal*, weil es von station ren Bedingungen ausgeht, d.h. es wird von einem gr  tenteils konstant durchflossenen Rohr mit konstanten Rand- und Umgebungsbedingungen und konstanter Vorlauftemperatur ausgegangen. Entlang der Rohrschlange wird von konstanten Umgebungsbedingungen ausgegangen (es wird die Temperatur der aktiven Schicht verwendet).



Die Randbedingungen und Vorlauftemperatur k nnen sich zwar w hrend der Simulation  ndern, jedoch sind die Gleichungen streng genommen nur f r komplett station re Prozesse g ltig.

```
<IdealPipeRegisterModel id="701" displayName="Floor heating" modelType="Constant">
  <ThermostatZoneId>1</ThermostatZoneId>
  <ConstructionObjectList>Floor</ConstructionObjectList>
  <IBK:Parameter name="SupplyTemperature" unit="C">8</IBK:Parameter>
  <IBK:Parameter name="MaxMassFlux" unit="kg/s">0.2</IBK:Parameter>
  <IBK:Parameter name="PipeLength" unit="m">100</IBK:Parameter>
  <IBK:Parameter name="PipeInnerDiameter" unit="mm">25.6</IBK:Parameter>
  <IBK:Parameter name="UValuePipeWall" unit="W/mK">5</IBK:Parameter>
  <!-- The default value for number of pipes is 1, so this could be omitted. -->
  <IBK:IntPara name="NumberParallelPipes">1</IBK:IntPara>
  <!-- Fluid properties of water -->
  <HydraulicFluid displayName="Water">
    <IBK:Parameter name="Density" unit="kg/m3">998</IBK:Parameter>
    <IBK:Parameter name="HeatCapacity" unit="J/kgK">4180</IBK:Parameter>
    <IBK:Parameter name="Conductivity" unit="W/mK">0.6</IBK:Parameter>
    <LinearSplineParameter name="KinematicViscosity" interpolationMethod="Linear">
      <X unit="C">0 90 </X>
      <Y unit="m2/s">1.307e-06 1.307e-06</Y>
    </LinearSplineParameter>
  </HydraulicFluid>
</IdealPipeRegisterModel>
```

Ein **IdealPipeRegisterModel** kann die Heiz- und K hleleistung in mehreren Konstruktionen berechnen. Daf r identifiziert der Objektlisten-tag **ConstructionObjectList** die Konstruktion, welche durch das Modell beheizt werden. Jede beheizte Konstruktion darf nur von einem Modell angesprochen werden.

- Die in der Objektliste referenzierten Konstruktionen m ssen einen Konstruktionstyp haben, in dem eine aktive Schicht definiert ist (siehe [Aktive/beheizte Schichten](#)).

Die Heizleistung wird  ber die Anpassung des Massestroms in Abh ngigkeit eines Thermostat-Kontrollsignals gesteuert. Die Zone, in der sich das Thermostat befindet, wird im Element **ThermostatZoneId** angegeben.

Das **IdealPipeRegisterModel** unterst tzt die folgenden XML-Attribute:

Tabelle 36. Attribute

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Kennung des Modells	int	<i>erforderlich</i>
<b>displayName</b>	Anzeigename/Beschreibung	string	<i>optional</i>
<b>modelType</b>	Definiert die Vorlauftemperatur <ul style="list-style-type: none"> <li>� <b>Constant</b> - Konstante Vorlauftemperatur</li> <li>� <b>Scheduled</b> - Vorlauftemperatur �ndert sich nach einem definierten Zeitplan.</li> </ul>	key	<i>erforderlich</i>



Wenn der Modelltyp **Scheduled** verwendet wird, sind folgende Zeitplanparameter erforderlich:

¥ **SupplyTemperatureSchedule** [C]

¥ **MaxMassFluxSchedule** [kg/s]

Diese müssen für die Konstruktion definiert werden. Häufig ist es sinnvoll, die gleiche Objektliste für die Schedule-Definition wie auch für das **IdealPipeRegisterModel** zu verwenden.

Folgende Fließkommaparameter (siehe **IBK:Parameter** für eine Beschreibung des Elementtyps **IBK:Parameter**) sind anzugeben:

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
<b>SupplyTemperature</b>	C	Vorlauftemperatur	$\hat{T} \geq -100.0 \hat{T}$	<i>erforderlich nur für Modelltyp <b>pConstant</b></i>
<b>MaxMassFlux</b>	kg/s	Maximaler Massestrom durch das gesamte Rohrregister	$\hat{E} = \hat{D}.0 \hat{E}$	<i>erforderlich nur für Modelltyp <b>pConstant</b></i>
<b>PipeLength</b>	m	Länge eines Rohrs im Rohrregister	$\hat{E} = \hat{D}.0 \hat{E}$	<i>erforderlich</i>
<b>PipeInnerDiameter</b>	m	Innendurchmesser des Rohrs	$\hat{E} = \hat{D}.0 \hat{E}$	<i>erforderlich</i>
<b>UValuePipeWall</b>	W/mK	Längenbezogener äquivalenter U-Wert der Rohrwand	$\hat{E} = \hat{D}.0 \hat{E}$	<i>erforderlich</i>

Der Parameter **NumberParallelPipes** gibt an, wie viele Rohre parallel in der Konstruktion durchströmt werden. Der insgesamt durchströmte Querschnitt ergibt somit aus dem Querschnitt eines Rohres mal Anzahl parallel durchströmter Rohre.

Diese Angabe hat Einfluss auf die Berechnung der Strömungsgeschwindigkeit, da der Massenstrom durch das Rohrregister gleichmäßig auf alle parallelen Stränge aufgeteilt wird. Dies hat auch Auswirkung auf den Begrenzungsparameter **MaxMassFlux**. Ist dieser bspw. auf 0,1 kg/s eingestellt, und es sind zwei parallele Rohrregister verbaut (**NumberParallelPipes=2**), so wird jedes der parallel verlegten Rohre mit max. 0,05 kg/s durchströmt.

!

Wenn man eine einzelne Rohrschlange in einer Konstruktion modelliert, dann gibt man die Länge der Rohrschlange und als **NumberParallelPipes** 1 an. Verlegt man mehrere Rohrschlangen, bzw. hat parallele Rohre, dann gibt man als Rohrlänge jeweils

die Länge eines einzelnen Rohres an und die Anzahl der Rohrschlangen/parallel verlegten Rohre.

### 17.10.1. Heiz- und K hleistung

Das Regelsignal `HeatingControlValue` wird als Wert zwischen 0..1 interpretiert. Werte au erhalb dieser Grenzen werden abgeschnitten. Bei 1 wird der maximale Massestrom f r Heizung auf den Wert `MaxMassFlow` gesetzt. Bei 0 ist die Heizung aus (Massestrom = 0). Dazwischen wird linear interpoliert.

Die K hlung ist genauso definiert. Bei einem Kontrollwert von 1 l uft die K hlung mit maximaler Kraft, bei 0 ist sie aus.

!!

Grunds tzlich kann ein Rohrregister nur W rme abgeben, wenn die Vorlauftemperatur h her als die Schichttemperatur ist. Daher wird auch bei einer Heizanforderung durch das Thermostat der Massestrom auf 0 gesetzt, wenn die Vorlauftemperatur zu klein wird.

Dadurch ist es auch niemals m glich, dass sowohl Heizung als auch K hlung berechnet wird. Also selbst wenn das Thermostat sowohl Heiz- als auch K hlanforderung gibt, kann es immer nur entweder Heizung oder K hlung geben, je nach Vorlauf- und Schichttemperatur.

### 17.10.2. Ausgaben

Ergebnisgr  en des Modells sind:

  `MassFlow` [kg/s]

  `ActiveLayerThermalLoad` [W]

Diese konstruktionsspezifischen Gr  en werden als vektorwertige Ergebnisgr  en bereitgestellt, mit Zugriff  ber die ID der Konstruktionsinstanz: Z. B. ist `Model<IdealSurfaceHeatingCoolingModel_id>.ActiveLayerThermalLoad[id=3]` die Heizlast in Konstruktion #3.

!

Die `ActiveLayerThermalLoad` ist positiv bei Beheizung der Fl sche und negativ bei K hlung. Die W rme-/K hleleistung ist eine absolute Gr  e (nicht fl schenbezogen).

## 17.11. Modell f r die Summation von Heiz- und K hleleistungen

Das Modell summiert die Heizleistungen der ausgew hlten Objekte und liefert die Gesamtw rmeleistung (Heizung/K hlung).

*Beispiel 63. Definitionsblock f r ein Summationsmodell*

```
<HeatLoadSummationModel id="801" displayName="Floor heating loads">
    <ObjectList>Floors</ObjectList>
```

```
</HeatLoadSummationModel>
```

Das `HeatLoadSummationModel` muss mit den folgenden XML-Attributen definiert werden:

Tabelle 37. Attribute

Attribut	Beschreibung	Format	Verwendung
<code>id</code>	Kennung des Modells	<code>int</code>	<i>erforderlich</i>
<code>displayName</code>	Anzeigename/Beschreibung	string	<i>optional</i>
<code>useZoneCoolingLoad</code>	für die Summation idealer Zonenlasten, <i>true</i> für Kühltlasten, <i>false</i> für Heizlasten (default) = <i>false</i>	string	<i>optional</i>

Ein `HeatLoadSummationModel` summiert die idealen Heiz- oder Kühltlasten mehrerer Räume, die Wärmelasten in Konstruktionen oder in hydraulische Netzwerke. Nur einer der Typen `Zone`, `ConstructionInstance`, `NetworkElement` ist dabei im Modell erlaubt. Dafür identifiziert der Objektlisten-tag `ObjectList` den Objekttyp und die Auswahl von Räumen/Konstruktionen/Netzwerkelementen, deren Heizlast erfasst und summiert werden soll. Im Fall der idealen Raumbeheizung oder Raumkühlung ist durch die Option `useZoneCoolingLoad` die passende Lastart (Heizung oder Kühlung) auszuwählen.



Falls in der Objektliste referenzierte Konstruktionen keine aktive Schicht definiert haben (siehe [Aktive/beheizte Schichten](#)), so werden diese einfach ignoriert.

#### 17.11.1. Ausgaben

Das Modell liefert die Summe der thermischen Lasten in der Ergebnisgröße `Total Thermal Load`.



Die `Total Thermal Load` ist positiv bei Beheizung von Räumen, Flächen und negativ bei Kühlung. Innerhalb von Netzwerken entspricht eine positive Last der Wärmeaufnahme des Fluides und negativ der Wärmeabgabe im Fluid. Im stationären Fall entspricht die `Total Thermal Load` in allen aktiv geheizten Netzwerkelementen also exakt der `Total Thermal Load` in die geheizten Konstruktionen.

### 17.12. Schnittstellen-Modell für die Anbindung externer Anlagennetze

Dieses Modell ist für alleinstehende Simulationen unwichtig und wird primär für die Co-Simulation mittels FMI gebraucht. Es bildet den Wärmeentzug des Gebäudes auf ein strömendes Medium ab, bspw. ein Versorgungsnetzwerk. Das Medium strömt mit gegebener Vorlauftemperatur und Massenstrom ins Gebäude und das `NetworkInterfaceAdapterModel` berechnet die resultierende Rücklauftemperatur.

Vorlauftemperatur und Massenstrom werden als Zeitprofile benötigt:

✶ `SupplyTemperatureSchedule [C]`

⚠ **MassFluxSchedule** [kg/s]

Bei Verwendung der FMI-Kopplung sind diese Größen durch FMI-Input-Variablen zu überschreiben.

Beispiel 64. Definitionsblock für ein solches Schnittstellenmodell

```
<NetworkInterfaceAdapterModel id="802" summationModelId="801">
  <IBK:Parameter name="FluidHeatCapacity" unit="J/kgK">4180</IBK:Parameter>
</NetworkInterfaceAdapterModel>
```

Das **NetworkInterfaceAdapterModel** muss mit den folgenden XML-Attributen definiert werden:

Tabelle 38. Attribute

Attribut	Beschreibung	Format	Verwendung
id	Kennung des Modells	int	erforderlich
displayName	Anzeigename/Beschreibung	string	optional
summationModelId	ID des Summationsmodells, welches die Heiz-/Kühlleistung im Gebäude berechnet	int	erforderlich

Für die Berechnung ist die spezifische Wärmekapazität des strömenden Mediums notwendig, angegeben im IBK-Parameterelement **FluidHeatCapacity**.

17.12.1. Ausgaben

Das Modell hat eine einzige Ergebnisgröße:

⚠ **ReturnTemperature** [C]



Der Wärmeentzug aus dem Versorgungsstrang erfolgt unabhängig eventueller physikalischer Grenzen. In der Realität die Rücklauftemperatur niemals unterhalb der Raum-/Konstruktionstemperatur sinken. Bei Einsatz dieses Modells ist zwar die Energiebilanz stets erfüllt, aber die Rücklauftemperatur kann sehr niedrig werden. Entsprechend sollten nachfolgende Modelle diese Temperatur nur für Regelungen als Sensorgröße behandeln und ggfs. auf sinnvolle Wertebereiche begrenzen.

18. Thermo-hydraulische Netzwerke

Dieser Abschnitt beschreibt die Parametrisierung von thermo-hydraulischen Netzwerken.

NANDRAD unterstützt geschlossene hydraulische Netzwerke, d.h. typische Rohrnetzwerke, Fußbodenheizungen, Wärmetauscher etc. und offene Luftströmungsnetzwerke. In letzteren wird ein Luftaustausch mit Raumluftzonen möglich.

In einer Projektdatei kann es mehrere Netzwerke geben. Diese sind im XML-tag **HydraulicNetworks** abgelegt.

#### Beispiel 65. Hydraulische Netzwerke

```
<HydraulicNetworks>
  <HydraulicNetwork id="1" displayName="xxx">
    ... <!-- network parameters -->
  </HydraulicNetwork>

  <HydraulicNetwork id="2" displayName="yyy">
    ... <!-- network parameters -->
  </HydraulicNetwork>

  ... <!-- other networks -->

</HydraulicNetworks>
```

■

Die einzelnen Netzwerke sind hydraulisch unabhängig, d.h. es gibt keinen Fluidaustausch zwischen Netzwerken. Die Netzwerke können aber thermisch verknüpft sein, z.B. kann ein Quartierswärmernetz mit einem Gebäudeanlagenetz Wärme austauschen. Dies bedingt auch die *eindeutige ID* der in *allen* Netzwerken definierten hydraulischen Netzwerkkomponenten.

## 18.1. Definition eines hydraulischen Netzwerks

Ein einzelnes Netzwerk wird im XML-tag **HydraulicNetwork** definiert, wie im folgenden Beispiel:

#### Beispiel 66. Typische Definition eines einzelnen Netzwerks

```
<HydraulicNetwork id="1" displayName="Simple network" modelType="ThermalHydraulicNetwork"
  referenceElementId="201">
  <HydraulicFluid displayName="Water">
    ... <!-- fluid parameters -->
  </HydraulicFluid>

  <PipeProperties>
    ... <!-- database with pipe definitions -->
  </PipeProperties>

  <Components>
    ... <!-- database with component definitions -->
  </Components>

  <Elements>
    ... <!-- network flow elements with topology definition -->
  </Elements>

  <ControlElements>
    ... <!-- network flow controller parameters -->
  </ControlElements>

</HydraulicNetwork>
```

Der XML-tag **Hydraul i cNetwork** hat folgende Attribute:

Tabelle 39. Attributes

Attribut	Beschreibung	Format	Verwendung
<b>i d</b>	Eindeutige Identifizierungsnummer	ℰ-ℰℰ	<i>benötigt</i>
<b>di spl ayName</b>	Anzeigename/Beschreibung des Netzwerks	string	<i>optional</i>
<b>model Type</b>	Legt das Berechnungsmodell fest	string	<i>optional</i> (defaults to <b>Thermal Hydraul i cNetwork</b> )
<b>referenceEl ementI d</b>	ID des Referenzelements	ℰ-ℰℰ	<i>benötigt</i>

Das optionale Attribut **model Type** beschreibt, welches Berechnungsmodell zu verwenden ist ( [Tabelle 40](#)).

Tabelle 40. Werte für Attribut **model Type**

Attribut	Beschreibung
<b>Hydraul i cNetwork</b>	Nur die hydraulische Berechnung (Masseströme und Drücke) wird durchgeführt.
<b>Thermal Hydraul i cNetwork</b>	Netzwerk wird mit eingeschalteten Energiebilanzen berechnet. Jedes Strömungselement entspricht mindestens einer Energiebilanzgleichung.

Die Wahl des Netzwerkberechnungsmodells hat Auswirkungen auf die benötigte Parametrisierung der beteiligten Komponenten.

18.1.1. Parameter

Je nach Modelltyp können mehrere globale Netzwerkparameter definiert werden (siehe Abschnitt [IBK:Parameter](#) für eine Beschreibung des **IBK:Parameter** tags):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<b>Defaul tFl ui dTemperature</b>	C	einheitliche, konstante Fluidtemperatur zur Verwendung mit einem rein hydraulischen Modell	ℰ-ℰ0.0ℰ	<i>benötigt für Modelltyp <b>Hydraul i cNetwork</b></i>
<b>I ni ti al Fl ui dTemperature</b>	C	Anfangsfluidtemperatur	ℰ-ℰ0.0ℰ	<i>benötigt für Modelltyp <b>Thermal Hydraul i cNetwork</b></i>
<b>ReferencePressure</b>	Pa	Druck hinter dem Referenzelement	ℰ-ℰ0.0ℰ	<i>optional</i> (Standardwert ist 0 Pa)

Innerhalb des Netzwerk-Definitionsblocks gibt es vier Kind-Elemente:

- ¥ **HydraulicFluid** - definiert Medieneigenschaften
- ¥ **PipeProperties** - Rohreigenschaften
- ¥ **Components** - Komponentendefinitionen
- ¥ **Elements** - Netzwerkelemente und Netzwerktopologie
- ¥ **ControlElements** - Massenstromregler

## 18.2. Medieneigenschaften

Der XML-tag **HydraulicFluid** enthält die Definition des im Netzwerk strömenden Mediums.



In jedem Netzwerk gibt es nur ein einziges Fluid, und deshalb ist auch nur eine einzige Instanz des **HydraulicFluid** Tags erlaubt.

### Beispiel 67. Definition eines Netzwerkfluids

```
<HydraulicFluid displayName="Water">
  <IBK:Parameter name="Density" unit="kg/m3">998</IBK:Parameter>
  <IBK:Parameter name="HeatCapacity" unit="J/kgK">4180</IBK:Parameter>
  <IBK:Parameter name="Conductivity" unit="W/mK">0.6</IBK:Parameter>
  <LinearSplineParameter name="KinematicViscosity" interpolationMethod="Linear">
    <X unit="C">0 10 20 30 40 50 60 70 80 </X>
    <Y unit="m2/s">1.793e-06 1.307e-06 1.004e-06 8.01e-07 6.58e-07 5.54e-07 4.75e-07 4.13e-07 3.65e-07 </Y>
  </LinearSplineParameter>
</HydraulicFluid>
```

Der XML-tag **HydraulicFluid** hat folgende Attribute:

Tabelle 41. Attributes

Attribut	Beschreibung	Format	Verwendung
<b>displayName</b>	Beschreibung des Fluids	string	<i>optional</i>

Parameter des Netzwerkfluids (siehe Abschnitt **IBK:Parameter** für eine Beschreibung des **IBK:Parameter** tags):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<b>Density</b>	kg/m3	Dichte bei Referenztemperatur	$\hat{E} \hat{D}.0 \hat{E}$	<i>benötigt</i>
<b>HeatCapacity</b>	J/kgK	Spezifische Wärmekapazität	$\hat{E} \hat{D}.0 \hat{E}$	<i>benötigt</i>
<b>Conductivity</b>	W/mK	Wärmeleitfähigkeit bei Referenztemperatur	$\hat{E} \hat{D}.0 \hat{E}$	<i>benötigt</i>

!

Die obigen Eigenschaften, insbesondere die Dichte, werden zur Vereinfachung als temperaturunabhängig konstant angenommen. Für die meisten Anwendungsfälle der thermo-hydraulischen Simulation im Gebäude-/Quartierskontext wird die thermische Ausdehnung des Fluids nicht benötigt. Und die Auslegung des Ausdehngefäßes erfolgt nicht mit der Simulation.

Desweiteren gibt es noch temperaturabhängige Parameter, welche in linear interpolierten Datentabellen abgelegt werden (siehe Abschnitt [LinearSplineParameter](#) für eine Beschreibung des [LinearSplineParameter](#) Elements):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
KinematicViscosity	m <sup>2</sup> /s	Kinematische Viskosität	0.0	benötigt

### 18.3. Rohreigenschaften

Die Rohreigenschaften legen die physikalische/geometrischen Eigenschaften eines Rohrtyps fest. Diese werden im XML-tag [HydraulicNetworkPipeProperties](#) im Katalog [PipeProperties](#) mit eindeutigen IDs aufgelistet.

#### Beispiel 68. Definition von Rohreigenschaften

```
<PipeProperties>
  <HydraulicNetworkPipeProperties id="1">
    <IBK:Parameter name="PipeRoughness" unit="mm">0.07</IBK:Parameter>
    <IBK:Parameter name="PipeInnerDiameter" unit="mm">25.6</IBK:Parameter>
    <IBK:Parameter name="PipeOuterDiameter" unit="mm">32</IBK:Parameter>
    <IBK:Parameter name="UValuePipeWall" unit="W/mK">5</IBK:Parameter>
  </HydraulicNetworkPipeProperties>

  ...
</PipeProperties>
```

Rohreigenschaften werden über das Attribut [pipePropertyId](#) eines Netzwerkelements (siehe [Abschnitt 18.5](#)) referenziert.

Tabelle 42. Attribute

Attribut	Beschreibung	Format	Verwendung
id	Eindeutige Identifikationsnummer des Rohrdatensatzes	0	benötigt

Parameter der Rohreigenschaften (siehe Abschnitt [IBK:Parameter](#) für eine Beschreibung des [IBK:Parameter](#) Tags):



Name	Einheit	Beschreibung	Wertebereich	Verwendung
PipeRoughness	mm	Rauhheit der inneren Rohroberfläche	0.0	benötigt
PipeInnerDiameter	mm	Innendurchmesser des Rohres	0.0	benötigt
PipeOuterDiameter	mm	Außendurchmesser des Rohres	0.0	benötigt
UValuePipeWall	W/mK	Längenbezogener äquivalenter U-Wert der Rohrwand (einschließlich Dämmung, wenn vorhanden)	0.0	benötigt (für Rohre mit Wärmeleitung nach Außen)

Der Außendurchmesser muss größer als der Innendurchmesser sein.

Der längenbezogene äquivalente U-Wert der Rohrwand (einschließlich möglicher Dämmung) ist in der Berechnung so definiert, dass eine Multiplikation mit der Temperaturdifferenz zwischen Fluidtemperatur und Außentemperatur zum Wärmestrom pro m Rohrlänge führt. D.h. bei der Berechnung dieses äquivalenten U-Werts müssen Zylinderkoordinaten berücksichtigt werden. Der tatsächlichen Wärmestrom von Fluid zu Umgebung wird noch durch Übergangskoeffizienten (siehe u.A. Abschnitt [Abschnitt 18.4.1](#)) beeinflusst.

## 18.4. Komponentendefinitionen

Eine `HydraulicNetworkComponent` definiert die Basiseigenschaften eines Strömungselements. Diese werden in dem Katalog `Components` mit eindeutigen IDs aufgelistet.

*Beispiel 69. Definition einer Komponente*

```
<Components>
  <HydraulicNetworkComponent id="1" modelType="ConstantPressurePump">
    <IBK:Parameter name="PressureHead" unit="Pa">1000</IBK:Parameter>
    <IBK:Parameter name="Volume" unit="m3">0.01</IBK:Parameter>
  </HydraulicNetworkComponent>

  ...
</Components>
```

Tabelle 43. Attribute

Attribut	Beschreibung	Format	Verwendung
id	Eindeutige Identifikationsnummer der Komponente	0.0	benötigt

Attribut	Beschreibung	Format	Verwendung
model Type	Modelltyp	string	benötigt
displayName	Anzeigename/Beschreibung	string	optional

Die weiteren Parameter sind dann abhängig vom **model Type** der Komponente und dem **model Type** des Netzwerks.

#### 18.4.1. Modelltyp: SimplePipe

**SimplePipe** ist ein einfaches Rohrmodell, bei dem das gesamte Rohr als ein zusammenhängendes Fluidvolumen mit entsprechend gemittelten Eigenschaften beschrieben wird.

Für das Modell **SimplePipe** werden keine weiteren Parameter benötigt.

#### 18.4.2. Modelltyp: DynamicPipe

Die **DynamicPipe** ist ein detailliertes Rohrmodell, bei dem das Rohr entlang der Rohrlänge räumlich diskretisiert wird.

Es werden die folgenden Parameter benötigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
PipeMaxDiscretizationWidth	m	Länge der diskretisierten Elemente	>0	benötigt

#### 18.4.3. Modelltyp: ConstantPressurePump

Diese Komponente prägt eine konstante Druckdifferenz unabhängig vom Massenstrom auf. Das Modell bildet somit eine geregelte Pumpe mit konstanter Druckerhöhung ab. Für das Modell **ConstantPressurePump** werden diese Parameter benötigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
PressureHead	Pa	Konstante Druckhöhe, welche die Pumpe erzeugt	beliebig	
PumpEfficiency	-	Gesamtwirkungsgrad der Pumpe	0 ≤ 1, 1.0	benötigt für Modelltyp Thermal Hydraulic Network
Volume	m³	Fluid volume inside the pump	1.0	benötigt für Modelltyp Thermal Hydraulic Network

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>FractionOfMotorInefficienciesToFluidStream</code>	-	Anteil der ans Fluid abgegebenen WŠrmeverluste der Pumpe (Standardwert 100%, NasslŠufer)	0 $\leq$ 1, $\hat{E} \hat{D}.0$	<i>optional fŠr Modelltyp Thermal Hydraulic Network</i>
<code>MaximumPressureHead</code>	Pa	Maximale DruckhŠhe bei minimalem Massenstrom, wird fŠr die Berechnung der massenstromabhŠngigen maximalen DruckhŠhe verwendet	$\hat{E} \hat{D}.0$	<i>optional fŠr Modelltyp Thermal Hydraulic Network</i>
<code>PumpMaximumElectricalPower</code>	W	Maximale elektrische Leistung der Pumpe, wird fŠr die Berechnung der massenstromabhŠngigen maximalen DruckhŠhe verwendet	$\hat{E} \hat{D}.0$	<i>optional fŠr Modelltyp Thermal Hydraulic Network</i>

Der Parameter `PressureHead` legt eine konstante DruckhŠhe fest. Es ist jedoch auch mŠglich, eine Zeitreihe fŠr diesen Parameter im Zeitplanparameter `PressureHeadSchedule` anzugeben. Wird in solcher Zeitplan fŠr das jeweilige Pumpen-StrŠmungselement gefunden, so wird dieser Anstelle des konstanten Parameters verwendet.

Werden die optionalen Parameter `MaximumPressureHead` und `PumpMaximumElectricalPower` angegeben, so wird eine maximale DruckerhŠhung in AbhŠngigkeit des Massenstroms berechnet (linear) und die tatsŠchliche DruckerhŠhung dementsprechend begrenzt.



Im Gegensatz zu anderen Modellen in NANDRAD, bei denen explizit zwischen konstantem Parameter und ZeitplŠnen unterschieden wird, erfolgt bei diesem Modell die Auswahl nach VerfŠgbarkeit eines definierten Parameters. Dies birgt das Risiko, dass bei Eingabe-/Modellierungsfehlern (z.B. falsche Objektlisten) der Parameter `PressureHeadSchedule` nicht fŠr das Pumpenelement gefunden wird, und dies nicht zu einer Warnung/Fehlermeldung fŠhrt. Stattdessen wŠrde stillschweigend der konstante Parameter verwendet werden.

Da sich dies in den Simulationsergebnissen jedoch leicht erkennen lŠsst, sollten entsprechende Fehler doch recht einfach zu finden sein. Deshalb wird in diesem Modell auf ein zusŠtzliches Attribut zur Wahl der Parametrierung verzichtet.



Die ModellabhŠngigkeit von der Zeitplanvariable `PressureHeadSchedule` kann genutzt werden, um die DruckhŠhe als FMU-EingangsgrŠŠe zu definieren.

Die Pumpeneffizienz ist als der mechanische Gesamtwirkungsgrad der Pumpe definiert. D.h. die durch Volumenstrom und DruckhŠhe gegebene mechanische Arbeit entspricht diesem Anteil der Gesamtarbeit. Die Differenz der Leistungen wird anteilig entsprechend des Parameters `FractionOfMotorInefficienciesToFluidStream` als WŠrmequelle dem Fluid aufgeprŠgt.

#### 18.4.4. Modelltyp: VariablePressurePump

Das Modell bildet eine Pumpe mit linear ansteigender Druckerhöhung in Abhängigkeit des Massenstroms ab. Damit entspricht die Pumpenkennlinie einer sogenannten dp-v Regelung. Es werden die folgenden Parameter benötigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
PumpEfficiency	-	Gesamtwirkungsgrad der Pumpe	0 ≤ 1, $\hat{=}$ 0.0	benötigt für Modelltyp ThermalHydraulicNetwork
Volume	m <sup>3</sup>	Fluid volume inside the pump	$\hat{=}$ 0.0 $\hat{=}$	benötigt für Modelltyp ThermalHydraulicNetwork
DesignPressureHead	Pa	Druckerhöhung am Auslegungspunkt, benötigt für die Berechnung der linearen Kennlinie	$\hat{=}$ 0.0 $\hat{=}$	benötigt für Modelltyp ThermalHydraulicNetwork
DesignMassFlux	kg/s	Massenstrom am Auslegungspunkt, benötigt für die Berechnung der linearen Kennlinie	$\hat{=}$ 0.0 $\hat{=}$	benötigt für Modelltyp ThermalHydraulicNetwork
PressureHeadReduction	-	Mit diesem Faktor wird die Druckerhöhung bei einem Massenstrom von 0 berechnet, womit die lineare Kennlinie bestimmt wird. Dazu wird die Druckerhöhung am Auslegungspunkt mit diesem Faktor multipliziert.	0 ≤ 1, $\hat{=}$ 0.0 $\hat{=}$	benötigt für Modelltyp ThermalHydraulicNetwork
FractionOfMotorInefficienciesToFluidStream	-	Anteil der ans Fluid abgegebenen Wärmeverluste der Pumpe (Standardwert 100%, Nassläufer)	0 ≤ 1, $\hat{=}$ 0.0	optional für Modelltyp ThermalHydraulicNetwork
MaximumPressureHead	Pa	Maximale Druckhöhe bei minimalem Massenstrom, wird für die Berechnung der massenstromabhängigen maximalen Druckhöhe verwendet	$\hat{=}$ 0.0	optional für Modelltyp ThermalHydraulicNetwork
PumpMaximumElectricalPower	W	Maximale elektrische Leistung der Pumpe, wird für die Berechnung der massenstromabhängigen maximalen Druckhöhe verwendet	$\hat{=}$ 0.0	optional für Modelltyp ThermalHydraulicNetwork

Werden die optionalen Parameter **MaximumPressureHead** und **PumpMaximumElectricalPower** angegeben, so

wird eine maximale Druckerhöhung in Abhängigkeit des Massenstroms berechnet (linear) und die tatsächliche Druckerhöhung dementsprechend begrenzt. Die Pumpeneffizienz ist als der mechanische Gesamtwirkungsgrad der Pumpe definiert. D.h. die durch Volumenstrom und Druckhöhe gegebene mechanische Arbeit entspricht diesem Anteil der Gesamtarbeit. Die Differenz der Leistungen wird anteilig entsprechend des Parameters `Fracti onOfMotorI neffi ci enci esToFl ui dStream` als Wärmequelle dem Fluid aufgeprägt.

#### 18.4.5. Modelltyp: ConstantMassFluxPump

Analog zur idealen druckgeführten Pumpe prägt dieses Modell einen vorgegebenen Massenstrom auf.



Es ist sehr leicht mit diesen Pumpenelementen unlösbare Gleichungssysteme zu formulieren. Beispielsweise können die Netzwerkgleichungen bei zwei solcher Pumpenelemente in einem seriellen Kreis und unterschiedlichen Vorgabemassenströmen nicht gelöst werden. Bei der Modellierung muss dies entsprechend geprüft werden.

Für das Modell `ConstantMassFluxPump` werden diese Parameter benötigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>MassFlux</code>	kg/s	Vorgegebener Massenstrom durch die Pumpe	$>0$	
<code>PumpEfficiency</code>	-	Gesamtwirkungsgrad der Pumpe	$0 \leq 1, \neq 0.0$	<i>benötigt für Modelltyp ThermalHydraulicNetwork</i>
<code>Fracti onOfMotorI neffi ci enci esToFl ui dStream</code>	-	Anteil der ans Fluid abgegebenen Wärmeverluste der Pumpe (Standardwert 100%, Nussliufer)	$0 \leq 1, \neq 0.0$	<i>optional für Modelltyp ThermalHydraulicNetwork</i>
<code>Volume</code>	m <sup>3</sup>	Fluid volume inside the pump	$\neq 0.0$	<i>benötigt für Modelltyp ThermalHydraulicNetwork</i>

Wie beim Modelltyp `ConstantPressurePump` kann der konstante Massenstrom im Parameter `MassFlux` durch einen optional gegebenen Zeitplan `MassFluxSchedule` (siehe Erläuterung oben beim Modelltyp `ConstantPressurePump`).

Die Berechnung der Verlustwärme und elektrischen Leistung erfolgt analog zur `ConstantPressurePump`.

#### 18.4.6. Modelltyp: ControlledPump

Dieses Modell ist ähnlich der `ConstantPressurePump`, mit dem Unterschied, dass die Druckhöhe nicht fest vorgegeben ist, sondern geregelt wird. Dazu können verschiedene Regler verwendet werden (siehe auch

## Beispiel 70. Definition einer ControlledPump

```

<!-- Komponentendefinition ist ähnlich der normalen Pumpe -->
<HydraulicNetworkComponent id="1" displayName="Pump" modelType="ControlledPump">
  <IBK:Parameter name="PumpEfficiency" unit="---">1</IBK:Parameter>
  <IBK:Parameter name="Volume" unit="m3">0.01</IBK:Parameter>
  <IBK:Parameter name="MaximumPressureHead" unit="Pa">10000</IBK:Parameter>
  <IBK:Parameter name="PumpMaximumElectricalPower" unit="W">50</IBK:Parameter>
</HydraulicNetworkComponent>

...

<!-- Das dazugehörige Element muss ein Kontrollelement referenzieren -->
<HydraulicNetworkElement id="1" inletNodeId="100" outletNodeId="0" componentId="1" controlElementId="1"
  displayName="Pump" />

```

Für das Modell **ControlledPump** werden diese Parameter benötigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<b>PumpEfficiency</b>	-	Gesamtwirkungsgrad der Pumpe	0.1; 1.0	benötigt für Modelltyp Thermal Hydraulic Network
<b>FractionOfMotorInefficienciesToFluidStream</b>	-	Anteil der ans Fluid abgegebenen Wärmeverluste der Pumpe (Standardwert 100%, Nassläufer)	0.1; 1.0	optional für Modelltyp Thermal Hydraulic Network
<b>Volume</b>	m <sup>3</sup>	Fluid volume inside the pump	0.01; 1.0	benötigt für Modelltyp Thermal Hydraulic Network
<b>MaximumPressureHead</b>	Pa	Maximum pressure head at point of minimal mass flow	1000; 100000	benötigt für Modelltyp Thermal Hydraulic Network
<b>PumpMaximumElectricalPower</b>	W	Maximum electrical power at point of optimal operation	10; 1000	benötigt für Modelltyp Thermal Hydraulic Network

Aus der maximalen Druckhöhe bei minimalem Durchfluss (**MaximumPressureHead**) und der maximalen Leistung im optimalen Betriebspunkt (**PumpMaximumElectricalPower**) berechnet das Modell die tatsächliche maximale Druckhöhe in Abhängigkeit des aktuellen Massenstroms. Diese tatsächliche maximale Druckhöhe sinkt mit zunehmenden Massenstrom linear und ist somit in der Regel geringer als die maximale Druckhöhe bei minimalem Durchfluss (**MaximumPressureHead**).

Das jeweilige Strömungselement ([Abschnitt 18.5](#)) referenziert im Attribut `controlElementId` einen Regler (siehe auch [Abschnitt 18.8](#)). Dieser bestimmt das Regelverhalten der Pumpe.

#### 18.4.7. Modelltyp: ConstantPressureLossValve

Dieses Modell bildet ein Ventil ab, welches unabhängig vom Massenstrom einen konstanten Druckverlust erzeugt.

Für das Modell `ConstantPressureLossValve` werden diese Parameter benötigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>PressureLoss</code>	Pa	Vorgegebener Druckverlust des Ventils	$\geq 0$	
<code>Volume</code>	m <sup>3</sup>	Fluidvolumen des Ventils	$\in [0, \infty)$	<i>benötigt für Modelltyp ThermalHydraulicNetwork</i>

!

Es ist ebenfalls möglich den Druckverlust über einen Zeitplan festzulegen. Dazu kann die Zeitplanvariable `PressureLossSchedule` genutzt werden. In diesem Fall wird der festgelegte Parameter `PressureLoss` nicht berücksichtigt.

#### 18.4.8. Modelltyp: PressureLossElement

Mit diesem Modell kann ein beliebiges Druckverlustelement (z.B. T-Stück, Ventil, etc.) abgebildet werden, welches durch einen zeta-Wert und den Innendurchmesser beschrieben werden kann.

Für das Modell `PressureLossElement` werden diese Parameter benötigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>HydraulicDiameter</code>	mm	Äquivalenter hydraulischer Durchmesser (wird für die Berechnung des Strömungsquerschnitts und der Strömungsgeschwindigkeit benötigt)	$\in [0, \infty)$	<i>benötigt</i>
<code>PressureLossCoefficient</code>	---	Effektiver Druckverlustbeiwert (zeta-Wert)	$\in [0, \infty)$	<i>benötigt</i>
<code>Volume</code>	m <sup>3</sup>	Fluidvolumen im Wärmetauscher	$\in [0, \infty)$	<i>benötigt für Modelltyp ThermalHydraulicNetwork</i>

!

Es ist zusätzlich möglich auch Elemente zu beschreiben, welche sich in identischen parallelen Strömen befinden. Dazu wird in dem `HydraulicNetworkElement` (siehe

[Abschnitt 18.5](#)), welches ein [PressureLossElement](#) referenziert, der Parameter [NumberParallelElements](#) angegeben. Siehe dazu auch [Abschnitt 18.5.1.2](#)

#### 18.4.9. Modelltyp: ControlledValve

Ein Ventil, welches nach bestimmten Kriterien geregelt wird. Letztlich ist dies ein normales Druckverlustelement, bei dem der letztlich wirksame Druckverlust dynamisch geregelt wird.

F  r das Model [ControlledValve](#) werden diese Parameter ben  tigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<a href="#">HydraulicDiameter</a>	mm	��quivalenter hydraulischer Durchmesser (wird f��r die Berechnung des Str��mungsquerschnitts und der Str��mungsgeschwindigkeit ben��tigt)	$\hat{>0.0}$	<i>ben��tigt</i>
<a href="#">PressureLossCoefficient</a>	---	Effektiver Druckverlustbeiwert (zeta-Wert)	$\hat{>0.0}$	<i>ben��tigt</i>
<a href="#">Volume</a>	m <sup>3</sup>	Fluidvolumen im W��rmetauscher	$\hat{>0.0}$	<i>ben��tigt f��r Modelltyp ThermalHydraulicNetwork</i>

Ein durchstr  mtes Element, welches eine Komponente von Modelltyp [ControlledValve](#) verwendet, referenziert in der Regel (aber nicht zwingend) ein Kontrollelement (siehe [Abschnitt 18.8](#)). Diese Massenstromkontroller erh  hen den Basisdruckverlustbeiwert. Ohne ein Kontrollelement verh  lt sich das [ControlledValve](#) einfach wie eine normales, druckverlustbehaftetes Element im Str  mungsnetzwerk.

#### 18.4.10. ModellTyp: HeatExchanger

Das Model [HeatExchanger](#) ist ein einfacher W  rme  bertrager, welcher mit dem Fluid einen vorgegebenen W  rmestrom austauscht. Es werden diese Parameter ben  tigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<a href="#">HydraulicDiameter</a>	mm	��quivalenter hydraulischer Durchmesser (wird f��r die Berechnung des Str��mungsquerschnitts und der Str��mungsgeschwindigkeit ben��tigt)	$\hat{>0.0}$	<i>ben��tigt</i>
<a href="#">PressureLossCoefficient</a>	---	Effektiver Druckverlustbeiwert (zeta-Wert)	$\hat{>0.0}$	<i>ben��tigt</i>



Name	Einheit	Beschreibung	Wertebereich	Verwendung
Volume	m <sup>3</sup>	Fluidvolumen im W�rmetauscher	[-10.0]	ben�tigt f�r Modelltyp Thermal Hydraulic Network

#### 18.4.11. ModellTyp: HeatPumpVariableIdealCarnotSourceSide

Das Modell **HeatPumpVariableIdealCarnotSourceSide** ist eine ideale W rmpumpe mit variable Heizleistung und gegebener Carnot-Effizienz, welche die Quellenseite der W rmpumpe modelliert. Mit diesem Modell wird also der W rme bertr ger der kalten Seite (Verdampfer) der W rmpumpe als Teil des Netzwerks modelliert. Dem Fluid wird dort W rme entzogen. Die von der W rmpumpe erzeugte W rme an der warmen Seite (Kondensator) wird durch ein W rmeaustauschmodell ([Abschnitt 18.6](#)) als Randbedingung angegeben. Weiterhin muss die mittlere Kondensatortemperatur (d.h. der gesch tzte Mittelwert zwischen Ein- und Austrittstemperatur am Kondensator, z.B. 32.5 C) in Form eines Schedules mit der Bezeichnung **CondenserMeanTemperatureSchedule** angegeben werden.

Die Heizleistung wird durch den **HydraulicNetworkHeatExchange** mit dem modelType **HeatLossSplitCondenser** bestimmt.

Weiterhin werden diese Parameter ben tigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
HydraulicDiameter	mm	�quivalenter hydraulischer Durchmesser (wird f�r die Berechnung des Str�mungsquerschnitts und der Str�mungsgeschwindigkeit ben�tigt)	[-10.0]	ben�tigt
PressureLossCoefficient	---	Effektiver Druckverlustbeiwert (zeta-Wert)	[-10.0]	ben�tigt
Volume	m <sup>3</sup>	Fluidvolumen im W�rmetauscher	[-10.0]	ben�tigt f�r Modelltyp Thermal Hydraulic Network
CarnotEfficiency	---	Carnot-Faktor zur Berechnung des COP	[-10.0]	ben�tigt f�r Modelltyp Thermal Hydraulic Network
MaximumHeatingPower	W	Maximale Heizleistung (= maximaler W�rmestrom des Kondensators)	[-10.0]	ben�tigt f�r Modelltyp Thermal Hydraulic Network

■

Wenn die Differenz zwischen mittlerer Kondensatortemperatur und mittlerer

Verdampfertemperatur weniger als 4 K betrřgt oder die Verdampfertemperatur -30°C unterschreitet, wird eine Warnung ausgegeben und die Wřrmepumpe ist ausgeschaltet. Alle Ausgaben des Modells sind dann auf 0.0 gesetzt.

#### Beispiel 71. Definition einer HeatPumpVariableIdealCarnotSourceSide

```
<Components>
É <HydraulicNetworkComponent id="2" modelType="HeatPumpVariableIdealCarnotSourceSide">
É   <IBK:Parameter name="HydraulicDiameter" unit="mm">25.6</IBK:Parameter>
É   <IBK:Parameter name="PressureLossCoefficient" unit="-">5</IBK:Parameter>
É   <IBK:Parameter name="Volume" unit="m3">0.001</IBK:Parameter>
É   <IBK:Parameter name="CarnotEfficiency" unit="---">0.4</IBK:Parameter>
É   <IBK:Parameter name="MaximumHeatingPower" unit="W">4000</IBK:Parameter>
É </HydraulicNetworkComponent>
</Components>

<Elements>
É <HydraulicNetworkElement id="1" inletNodeId="2" outletNodeId="0" componentId="2" displayName="heatpump">
É   <HydraulicNetworkHeatExchange modelType="HeatLossSplineCondenser">
É     <LinearSplineParameter name="HeatLoss" interpolationMethod="Linear">
É       <TSVFile>${ProjectDirectory}/file/to/HeatFluxCondenser.csv</TSVFile>
É     </LinearSplineParameter>
É   </HydraulicNetworkHeatExchange>
É </HydraulicNetworkElement>
</Elements>
```

#### 18.4.12. ModellTyp: HeatPumpVariableIdealCarnotSupplySide

Das Modell `HeatPumpVariableIdealCarnotSupplySide` ist eine ideale Wřrmepumpe mit variable Heizleistung und gegebener Carnot-Effizienz, welche die Senkenseite der Wřrmepumpe modelliert. Mit diesem Modell wird also der Wřrmeřbertrager der warmen Seite (Kondensator) der Wřrmepumpe als Teil des Netzwerks modelliert. Die auf der kalten Seite gegebene Temperatur wird durch ein Wřrmeaustauschmodell ([Abschnitt 18.6](#)) als Randbedingung angegeben. Die Wřrmeabgabe der Wřrmepumpe wird ideal geregelt so dass immer die gegebene Kondensatoraustrittstemperatur (= Heizungsvorlauftemperatur, z.B. 35°C) erreicht wird. die Kondensatoraustrittstemperatur muss in Form eines Schedules mit der Bezeichnung `CondenserOutletSetpointSchedule` angegeben werden.

Die benötigten Parameter sind identisch mit dem Modell [Abschnitt 18.4.11](#) und in [\[Parameter\\_HeatPumpVariableIdealCarnotSourceSide\]](#) gegeben.

#### ModellTyp: HeatPumpVariableSourceSide

Das Modell `HeatPumpVariableSourceSide` ist eine ideale Wřrmepumpe mit variable Heizleistung, welche die Quellenseite der Wřrmepumpe modelliert. Mit diesem Modell wird also der Wřrmeřbertrager der kalten Seite (Verdampfer) der Wřrmepumpe als Teil des Netzwerks modelliert. Dem Fluid wird dort Wřrme entzogen. Die von der Wřrmepumpe erzeugte Wřrme an der warmen Seite (Kondensator) wird durch ein Wřrmeaustauschmodell ([Abschnitt 18.6](#)) als Randbedingung angegeben. Weiterhin muss die mittlere Kondensatortemperatur (d.h. der geschřtzte Mittelwert zwischen Ein- und Austrittstemperatur am Kondensator, z.B. 32.5°C) in Form eines Schedules mit der Bezeichnung `CondenserMeanTemperatureSchedule` angegeben werden. Im Gegensatz zum Modell

HeatPumpVariableCarnotSourceSide, wird der COP durch ein Polynom bestimmt. Es werden 6 Polynomkoeffizienten (PolynomCoefficients) für die Berechnung des COP in Form eines DataTable benötigt (siehe Beispiel 72).

Weiterhin werden diese Parameter benötigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
HydraulicDiameter	mm	Äquivalenter hydraulischer Durchmesser (wird für die Berechnung des Strömungsquerschnitts und der Strömungsgeschwindigkeit benötigt)	$\hat{E} \hat{D} .0 \hat{E}$	benötigt
PressureLossCoefficient	---	Effektiver Druckverlustbeiwert (zeta-Wert)	$\hat{E} \hat{D} .0 \hat{E}$	benötigt
Volume	m3	Fluidvolumen im Wärmetauscher	$\hat{E} \hat{D} .0 \hat{E}$	benötigt für Modelltyp ThermalHydraulicNetwork
MaximumHeatingPower	W	Maximale Heizleistung (= maximaler Wärmestrom des Kondensators)	$\hat{E} \hat{D} .0 \hat{E}$	benötigt für Modelltyp ThermalHydraulicNetwork

■

Wenn die Differenz zwischen mittlerer Kondensatortemperatur und mittlerer Verdampfertemperatur weniger als 4 K beträgt oder die Verdampfertemperatur  $-30^{\circ}\text{C}$  unterschreitet, wird eine Warnung ausgegeben und die Wärmepumpe ist ausgeschaltet. Alle Ausgaben des Modells sind dann auf 0.0 gesetzt.

#### Beispiel 72. Definition einer HeatPumpVariableSourceSide

```

<Components>
  <HydraulicNetworkComponent id="2" modelType="HeatPumpVariableSourceSide">
    <IBK:Parameter name="HydraulicDiameter" unit="mm">25.6</IBK:Parameter>
    <IBK:Parameter name="PressureLossCoefficient" unit="-">5</IBK:Parameter>
    <IBK:Parameter name="Volume" unit="m3">0.001</IBK:Parameter>
    <IBK:Parameter name="MaximumHeatingPower" unit="W">4000</IBK:Parameter>
    <PolynomCoefficients>
      COP: -7.46562450e+01 5.87360996e-01 -1.38056160e-03 -4.45166703e-03 1.68697149e-03 1.77084233e-03
    </PolynomCoefficients>
  </HydraulicNetworkComponent>
</Components>

<Elements>
  <HydraulicNetworkElement id="1" inletNodeId="2" outletNodeId="0" componentName="heatpump">
    <HydraulicNetworkHeatExchange modelType="HeatLossSingleCondenser">
      <LinearSingleParameter name="HeatLoss" interpolationMethod="Linear">
        <TSVFile>${ProjectDirectory}/file/to/HeatFluxCondenser.csv</TSVFile>
      </LinearSingleParameter>
    </HydraulicNetworkHeatExchange>
  </HydraulicNetworkElement>

```

### 18.4.13. Modelltyp: HeatPumpOnOffSourceSide

Dieses Modell stellt eine reale On/Off-Wärmepumpe dar, welche nur an oder ausgeschaltet werden kann. Die Leistung und Effizienz werden durch Polynome bestimmt. Das Modell bildet die Quellenseite der Wärmepumpe ab, es wird also der Wärmeübertrager der kalten Seite (Verdampfer) der Wärmepumpe als Teil des Netzwerks modelliert.

Es werden je 6 Polynomkoeffizienten (**PolynomCoefficients**) für die Berechnung der Wärmeabgabe des Kondensators und der elektrischen Leistung werden in Form eines **DataTable** benötigt (siehe [Beispiel 73](#)). Es wird ein **Schedule** für das Signal zum An- und Ausschalten der Wärmepumpe mit der Bezeichnung **HeatPumpOnOffSignalSchedule** und ein **Schedule** für die Kondensatoraustrittstemperatur mit der Bezeichnung **CondenserOutletSetpointSchedule** benötigt.

Weitere benötigte Parameter sind:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<b>HydraulicDiameter</b>	mm	Äquivalenter hydraulischer Durchmesser (wird für die Berechnung des Strömungsquerschnitts und der Strömungsgeschwindigkeit benötigt)	1.0-10.0	<i>benötigt</i>
<b>PressureLossCoefficient</b>	---	Effektiver Druckverlustbeiwert (zeta-Wert)	1.0-10.0	<i>benötigt</i>
<b>Volume</b>	m <sup>3</sup>	Fluidvolumen im Wärmetauscher	1.0-10.0	<i>benötigt für Modelltyp ThermalHydraulicNetwork</i>

#### Beispiel 73. Definition einer HeatPumpOnOffSourceSide

```
<Components>
  <HydraulicNetworkComponent id="2" modelType="HeatPumpOnOffSourceSide">
    <IBK:Parameter name="HydraulicDiameter" unit="mm">25.6</IBK:Parameter>
    <IBK:Parameter name="PressureLossCoefficient" unit="---">5</IBK:Parameter>
    <IBK:Parameter name="Volume" unit="m3">0.001</IBK:Parameter>
    <PolynomCoefficients>
      QdotCondensator: -2.39461360e+04 -2.62085728e+02 2.83610229e+02 -1.64340182e+00 1.82863445e+00
      1.89987100e-01;
      Pel: 2.53859559e+04 9.26237012e+01 -2.74451964e+02 9.77255316e-02 -2.20526410e-01 4.61200937e-01
    </PolynomCoefficients>
  </HydraulicNetworkComponent>
</Components>
...
<Elements>
  <HydraulicNetworkElement id="1003" inletNodeId="2" outletNodeId="0" componentId="2" displayName="heatpump">
  </HydraulicNetworkElement>
```

```

</Elements>
...
<Schedule type="AllDays">
  <DailyCycles>
    <!-- Condenser Temperature is usually 35°C for space heating and two times per day 50°C for domestic hot water-->
    <DailyCycle interpolation="Constant">
      <TimePoints>0, 6, 8, 18, 20</TimePoints>
      <Values>CondenserOutletSetpointSchedule [C]: 35, 55, 35, 55, 35;</Values>
    </DailyCycle>
    <DailyCycle interpolation="Constant">
      <TimePoints>0, 2, 8, 16, 22</TimePoints>
      <Values>HeatPumpOnOffSignalSchedule [---]: 0, 1, 0, 1, 0;</Values>
    </DailyCycle>
  </DailyCycles>
</Schedule>

```

#### 18.4.14. Modelltyp: IdealHeaterCooler

Dieses Modell ermöglicht das präzise Vorgeben einer Medientemperatur in einem Strömungselement und der Berechnung der für die Temperaturänderung benötigten Heiz-/Kühlenergie.

*Beispiel 74. Definition der verknüpften Strömungselemente (Knotennummerierung erfolgt implizit)*

```

<Components>
  <...>

  <HydraulicNetworkComponent id="3" displayName="ideal heater cooler" modelType="IdealHeaterCooler"/>
</Components>

<Elements>

  <...>

  <!-- The ideal heat exchanger element requires 'SupplyTemperatureSchedule' schedule parameter -->
  <HydraulicNetworkElement id="3" inletNodeid="101" outletNodeid="100" componentId="3" displayName="Perfect heater/cooler"/>

</Elements>

...

  <ScheduleGroup objectList="Ideal heat exchanger">
    <Schedule type="AllDays">
      <DailyCycles>
        <DailyCycle interpolation="Constant">
          <TimePoints>0 6 18</TimePoints>
          <Values>
            SupplyTemperatureSchedule [C]: 22 35 22
          </Values>
        </DailyCycle>
      </DailyCycles>
    </Schedule>
  </ScheduleGroup>

...

  <ObjectList name="Ideal heat exchanger">
    <FilterID>3</FilterID>
  </ObjectList>

```

```
Ê <ReferenceType>NetworkElement</ReferenceType>
Ê </ObjectList>
```

Es sind keine weiteren Parameter notwendig. Es wird der Zeitplanparameter **SupplyTemperatureSchedule** erwartet (siehe [Beispiel 74](#)).

## 18.5. Strömungselemente

Das eigentliche Netzwerk wird durch die Definition konkreter Strömungselemente aufgebaut. Diese sind untereinander durch Einlass- und Auslassknoten verknüpft.

Die tatsächlichen Strömungselemente des Netzwerks werden innerhalb des XML-tags **Elements** mit dem XML-tag **HydraulicNetworkElement** definiert.

*Beispiel 75. Definition der verknüpften Strömungselemente (Knotennummerierung erfolgt implizit)*

```
<Elements>
Ê <HydraulicNetworkElement id="1" inletNodeId="5" outletNodeId="6" componentId="1" pipePropertiesId="1">
Ê   <IBK:Parameter name="Length" unit="m">100</IBK:Parameter>
Ê </HydraulicNetworkElement>
Ê <HydraulicNetworkElement id="2" inletNodeId="6" outletNodeId="7" componentId="2">
Ê </HydraulicNetworkElement>
Ê ...
</Elements>
```

**HydraulicNetworkElement**-tags haben die folgenden Attribute:

*Tabelle 44. Attribute*

Attribut	Beschreibung	Format	Verwendung
<b>id</b>	Eindeutige Identifikationsnummer des Strömungselements	Ê-ÊÊ	<i>benötigt</i>
<b>displayName</b>	Anzeigename/Beschreibung (verwendet für Ausgaben)	string	optional
<b>inletNodeId</b>	ID des Einlassknotens	Ê-ÊÊ	<i>benötigt</i>
<b>outletNodeId</b>	ID des Einlassknotens	Ê-ÊÊ	<i>benötigt</i>
<b>componentId</b>	ID des referenzierten <b>HydraulicNetworkComponent</b>	Ê-ÊÊ	<i>benötigt</i>
<b>pipePropertiesId</b>	ID des referenzierten <b>HydraulicNetworkPipeProperties</b>	Ê-ÊÊ	optional ( <i>benötigt für Rohre</i> )

Attribut	Beschreibung	Format	Verwendung
<code>ObservedPressureDifferenceElementIds</code>	Datatable mit IDs der Strömungselemente, welche mit WorstpointController überwacht werden	<code>String</code>	optional (benötigt nur wenn ein <code>HydraulicNetworkControllerElement</code> mit Option <code>WorstpointController</code> referenziert wird)

■

Die ID eines `HydraulicNetworkElement` muss global eindeutig sein, d.h. Strömungselemente müssen netzwerkübergreifend mit einer eindeutigen ID bezeichnet werden. Komponenten-IDs/Rohreigenschaften-IDs müssen nur innerhalb eines Netzwerkes eindeutig sein.

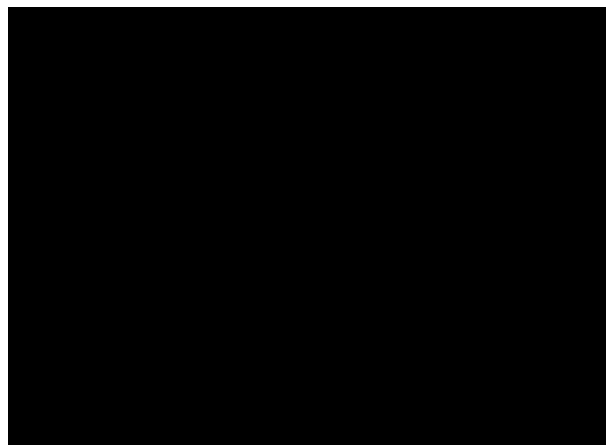


Abbildung 11. Einfaches Strömungsnetzwerk mit 3 Knoten und 3 Elementen

Die Strömungselemente sind miteinander durch Knoten verknüpft. In jedem Strömungselement fließt das Fluid (geplant) von dem Knoten mit der `inletNodeId` zu dem Knoten mit der `outletNodeId`. Während der Berechnung ist es jedoch möglich, dass sich der Massestrom umkehrt. Dies ändert aber nichts an der Topologiedefinition des Netzwerkes. Man könnte `inletNodeId` auch mit "Knoten 1 des Elements" und `outletNodeId` mit "Knoten 2 des Elements" bezeichnen.

Abbildung 11 zeigt ein einfaches Netzwerk bestehend aus 3 Elementen. Ein solches Netzwerk würde wie folgt definiert werden (Beispiel 76).

```
<Elements>
  <!-- Pump -->
  <HydraulicNetworkElement id="1" inletNodeId="1" outletNodeId="2" componentId="1"/>
  <!-- Pipe id=2-->
  <HydraulicNetworkElement id="2" inletNodeId="2" outletNodeId="3" componentId="2" pipePropertiesId="1">
    <IBK:Parameter name="Length" unit="m">10</IBK:Parameter>
  </HydraulicNetworkElement>
  <!-- Pipe id=3-->
  <HydraulicNetworkElement id="3" inletNodeId="3" outletNodeId="1" componentId="2" pipePropertiesId="1">
    <IBK:Parameter name="Length" unit="m">6</IBK:Parameter>
  </HydraulicNetworkElement>
</Elements>
```



Verschiedene Strömungselemente sind durch die Knoten IDs `inletNodeId` und `outletNodeId` verknüpft. Die Knoten-IDs referenzieren keine Strömungselemente, sondern "virtuelle" Knoten.

Jedes Strömungselement referenziert jeweils eine Komponente mit der `componentId`.

18.5.1. Rohr-Elemente

Ist eine Komponente ein Rohr (z.B. `Dynami cPipe`), müssen entsprechende Rohrparameter mit der `pipePropertiesId` referenziert werden.

Weiterhin muss für ein Rohrelement der Parameter `Length` definiert werden (siehe auch [Beispiel 77](#)):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>Length</code>	m	Rohrlänge	0.0	benötigt

Beispiel 77. Definition eines Rohrelements

```
<HydraulicNetworkElement id="2" inletNodeId="0" outletNodeId="1" componentId="3" pipePropertiesId="1">
  <IBK:Parameter name="Length" unit="m">100</IBK:Parameter>
</HydraulicNetworkElement>
```

Parallele Rohrregister (Flächenheizungen/Fußbodenheizung)

Durch Angabe des `IBK: IntPara` Tags mit dem Namen `NumberParallelPipes` kann man ein Rohrregister definieren und so z.B. eine Flächenheizung/Fußbodenheizung modellieren. Dabei strömt letztlich das Fluid parallel durch die angegebene Anzahl gleichartiger Rohre.



### Beispiel 78. Definition eines Rohrregisters

```
<HydraulicNetworkElement id="101" inletNodeId="2" outletNodeId="1" componentId="3" pipePropertiesId="1">
  <IBK:Parameter name="Length" unit="m">100</IBK:Parameter>
  <!-- We have a pipe register here, consisting of 10 parallel pipes -->
  <IBK:IntPara name="NumberParallelPipes">10</IBK:IntPara>
</HydraulicNetworkElement>
```



Die Ausgabe Massestrom **FluidMassFlow** und Volumenstrom **FluidVolumeFlow** beziehen sich auf das gesamte Rohrbündel.

### Parallele Druckverlustelemente (Ventile, Einbauten)

Durch Angabe des **IBK: IntPara** Tags mit dem Namen **NumberParallelElements** kann man ein Element mit Druckverlust definieren, welches identische in parallelen Strömen verbaut ist, ohne dazu jeden identischen Strang separat modellieren zu müssen.

### Beispiel 79. Definition eines parallelen Ventils

```
<HydraulicNetworkElement id="4" inletNodeId="5" outletNodeId="6" componentId="10020016" displayName="Ventil">
  <IBK: IntPara name="NumberParallelElements">2</IBK: IntPara>
</HydraulicNetworkElement>
```



Die Ausgabe Massestrom **FluidMassFlow** und Volumenstrom **FluidVolumeFlow** beziehen sich auf das gesamte Rohrbündel.

## 18.6. Wärmeaustauschmodelle

In thermische Netzwerken kann für ein Strömungselement ([Abschnitt 18.5](#)) ein Wärmeaustausch (mit der Umgebung, anderen Modellen, etc.) definiert werden. Dafür muss innerhalb der Definition des Strömungselements ein XML-Element **HydraulicNetworkHeatExchange** definiert werden.

### Beispiel 80. Definition von Strömungselementen mit **HydraulicNetworkHeatExchange**

```
<!-- Heat exchange with heat loss/gain pre-defined in time series -->
<HydraulicNetworkElement id="1" inletNodeId="1" outletNodeId="2"
  componentId="2" displayName="heat exchanger">

  <!-- Definition of pre-defined heat loss -->
  <HydraulicNetworkHeatExchange modelType="HeatLossSpline">
    <LinearSplineParameter name="HeatLoss" interpolationMethod="linear">
      <TSVFile>${Project Directory}/climate/HeatFlux.csv?2</TSVFile>
    </LinearSplineParameter>
  </HydraulicNetworkHeatExchange>
</HydraulicNetworkElement>

<!-- Pipe with heat exchange to the environment (with constant environment temperature) -->
```

```

<HydraulicNetworkElement id="2" inletNodeId="2" outletNodeId="3"
  componentId="3" pipePropertiesId="1" displayName="pipe">

  <IBK:Parameter name="Length" unit="m">100</IBK:Parameter>
  <!-- Definition of heat exchange with environment -->
  <HydraulicNetworkHeatExchange modelType="TemperatureConstant">
    <IBK:Parameter name="ExternalHeatTransferCoefficient" unit="W/m2K">5</IBK:Parameter>
    <IBK:Parameter name="Temperature" unit="C">0</IBK:Parameter>
  </HydraulicNetworkHeatExchange>
</HydraulicNetworkElement>

```

Es gibt verschiedene Modelltypen für den Wärmeaustausch, angegeben über das Attribut **modelType**:

modelType	Beschreibung	Verwendbar für Komponente
TemperatureConstant	Wärmeaustausch basierend auf Temperaturunterschied zwischen Medium und Umgebungstemperatur; Konstante Umgebungstemperatur ist als Parameter <b>Temperature</b> im <b>HydraulicNetworkElement</b> gegeben. Es muss zusätzlich der Parameter <b>ExternalHeatTransferCoefficient</b> gegeben sein.	SimplePipe, DynamicPipe
TemperatureSpline	Wärmeaustausch basierend auf Temperaturunterschied zwischen Medium und Umgebungstemperatur; Umgebungstemperatur ist als Zeitreihe in einem LinearSplineParameter ( <a href="#">Abschnitt 3.5</a> ) mit Namen <b>Temperature</b> gegeben. Es muss zusätzlich der Parameter <b>ExternalHeatTransferCoefficient</b> gegeben sein.	SimplePipe, DynamicPipe
TemperatureSplineEvaporator	Wärmeaustausch via Wärmepumpe, basierend auf gegebener Verdampfertemperatur. Verdampfertemperatur ist als Zeitreihe in einem LinearSplineParameter ( <a href="#">Abschnitt 3.5</a> ) mit Namen <b>Temperature</b> gegeben	HeatPumpIdealCarnot
TemperatureZone	Wärmeaustausch mit einer Zone. Raumlufttemperatur der referenzierten Zone ( <b>ZoneId</b> ) wird als Umgebungstemperatur verwendet. Es muss zusätzlich der Parameter <b>ExternalHeatTransferCoefficient</b> gegeben sein.	SimplePipe, DynamicPipe
TemperatureConstructionLayer	Wärmeaustausch mit einer Konstruktionsschicht (Fußbodenheizung/Flächenheizung); Schichttemperatur (aktive Schicht) der referenzierten Konstruktion ( <b>ConstructionInstanceId</b> ) wird als Umgebungstemperatur verwendet.	SimplePipe, DynamicPipe

model Type	Beschreibung	Verwendbar für Komponente
HeatLossConstant	Konstanter Wärmestrom (positiv aus dem Element) ist als konstanter Parameter <b>HeatLoss</b> gegeben.	SimplePipe, DynamicPipe, HeatExchanger
HeatLossSpline	Wärmestrom (positiv aus dem Element) ist als Zeitreihe in einem LinearSplineParameter ( <a href="#">Abschnitt 3.5</a> ) mit Namen <b>HeatLoss</b> gegeben	SimplePipe, DynamicPipe, HeatExchanger
HeatLossSplineCondenser	Wärmestrom (positiv aus dem Element) des Kondensators bei Verwendung eines Wärmepumpen-Modells als Zeitreihe in einem LinearSplineParameter ( <a href="#">Abschnitt 3.5</a> ) mit Namen <b>HeatLoss</b> gegeben	HeatPumpIdealCarnot



Wenn des XML-Element **HydraulicNetworkHeatExchange** fehlt, wird die entsprechende Komponente als adiabat behandelt.

### 18.6.1. Parameter für Wärmeaustauschdefinition

Konstante Parameter (Tag: **IBK:Parameter**):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Temperature	K	konstante Temperatur	$[-200.0; 200.0]$ °C	TemperatureConstant
ExternalHeatTransferCoefficient	W/m <sup>2</sup> K	äußerer Wärmeübergangskoeffizient	$[0.0; \infty]$	TemperatureConstant, TemperatureSpline, TemperatureZone
HeatLoss	W	konstante Wärmeabgabe	---	HeatLossConstant

Spline-Parameter (Tag: **LinearSplineParameter**):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Temperature	K	Zeitreihe mit Temperaturen	---	TemperatureSpline
HeatLoss	W	Zeitreihe mit Wärmeverlustströmen	---	HeatLossSpline, HeatLossSplineCondenser

ID-Referenzen:

XML-Tag-Name	Beschreibung	Verweis auf Datentyp	Verwendung
ZoneId	Referenz zur Zone	Zone	TemperatureZone
ConstructionInstanceId	Referenz zu beheizter/gekühlter Konstruktion	ConstructionInstance	TemperatureConstructionLayer

## 18.7. Regelung von Strömungselementen

Strömungselemente können ein geregeltes Ventil enthalten. Das Ventil wird z.B. so geregelt, dass eine vorgegebene Temperaturdifferenz oder ein vorgegebener Massenstrom erreicht wird. Im [Beispiel 81](#) referenziert das `HydraulicNetworkElement` dafür ein `HydraulicNetworkControlElement` im Attribut `controlElementId`.

*Beispiel 81. Beispiel für ein Wärmeübertrager mit geregelter Temperaturdifferenz*

```
<HydraulicNetworkElement id="2" inletNodeId="0" outletNodeId="101" componentId="2"
  Ê controlElementId="1" displayName="Heat exchanger">
  Ê <HydraulicNetworkHeatExchange modelType="HeatLossSpline">
  Ê   <LinearSplineParameter name="HeatLoss">
  Ê     <X unit="d">0 1 2 2.2 2.3 2.7 2.8 3</X>
  Ê     <Y unit="W">0 500 1000 1000 3000 3000 800 0</Y>
  Ê   </LinearSplineParameter>
  Ê </HydraulicNetworkHeatExchange>

</HydraulicNetworkElement>
...
<ControlElements>
  Ê <HydraulicNetworkControlElement id="1" controlledProperty="TemperatureDifference" modelType="Constant"
  Ê   controllerType="PController" >
  Ê   <IBK:Parameter name="Kp" unit="---">1e6</IBK:Parameter>
  Ê   <IBK:Parameter name="TemperatureDifferenceSetpoint" unit="K">3</IBK:Parameter>
  Ê   <!-- Deactivate upper limit of controller result value
  Ê     - this is the default and could be omitted -->
  Ê   <MaximumControllerResultValue>0</MaximumControllerResultValue>
  Ê </HydraulicNetworkControlElement>
</ControlElements>
```

Die konkrete Parametrierung des Kontrollers ist im referenzierten `HydraulicNetworkControlElement` enthalten. Die Massenstromregler arbeiten unterschiedlich und verlangen dabei wie im [Beispiel 81](#) eine entsprechende Wärmeaustauschparametrierung (`HydraulicNetworkHeatExchange`). Details zu diesen Abhängigkeiten sind in Abschnitt [Abschnitt 18.8](#) beschrieben.

## 18.8. Regler und Massenstromkontrollmodelle

Alle Reglerkomponenten werden in einer Liste `ControlElements` abgelegt, welche sich innerhalb des `HydraulicNetworkElement` befindet.

```
<ControlElements>
É <!-- This controller generates a control signal based on
É     temperature difference using a P-controller.
É     The temperature difference is computed from heat loss
É     and current mass flux. Hence, this controller can only
É     be used in a flow element that defines heat exchange
É     via prescribed heat loss.
É     This control element can be referenced by all flow
É     elements with similar mass flow control strategy.
É -->
É <HydraulicNetworkControlElement id="1" controlledProperty="TemperatureDifference" modelType="Constant"
controlledType="PController" >
É   <IBK:Parameter name="Kp" unit="---">1e6</IBK:Parameter>
É   <IBK:Parameter name="TemperatureDifferenceSetpoint" unit="K">3</IBK:Parameter>
É   <!-- Deactivate upper limit of controller result value
É       - this is the default and could be omitted -->
É   <MaximumControllerResultValue>0</MaximumControllerResultValue>
É </HydraulicNetworkControlElement>
</ControlElements>
```

Eine Reglerdefinition `HydraulicNetworkControlElement` beschreibt sowohl die Sensorgröße(n), das Berechnungsverfahren für das Reglereingangssignal und das Reglermodell selbst (P, PI-Regler, É). Das Ergebnis des Reglers ist ein zusätzlicher Druckverlustbeiwert zwischen 0 (kein zusätzlicher Strömungswiderstand) bis zu einem sehr hohen Wert (bspw. Ventil geschlossen).

Die Art der Regelung bzw. die zu Überwachende Größe wird im Attribut `controlledProperty` definiert. Das Attribut `modelType` (Optionen `Constant` oder `Scheduled`) definiert, ob die Setpoints als feste Parameter im `HydraulicNetworkControlElement` angegeben sind, oder als Zeitplanparameter angegeben sind.

### 18.8.1. Regler: TemperatureDifference

Dieser Regler kann nur bei Elementen mit vorgegebenem Wärmeverluststrom verwendet werden (Wärmeaustauschmodell `HeatLossConstant` oder `HeatLossSpline`). Aus dem gegebenen Wärmeverlust wird mit dem aktuellen Massenstrom der Temperaturabfall über dem Strömungselement berechnet. Dieser wird mit einem Sollwert (Parameter `TemperatureDifferenceSetpoint`) verglichen - die Differenz zwischen Sollwert und Ist-Temperaturdifferenz ist das Reglereingangssignal (Fehlerwert).

der Regler `TemperatureDifference` kann nur für das Modell `HeatExchanger` verwendet werden.

### 18.8.2. Regler: TemperatureDifferenceOfFollowingElement

Mit diesem Regler wird ein Ventil implementiert, welches die Temperaturdifferenz des nächsten Strömungselements (in Verknüpfungsreihenfolge) regelt. Das nächste Strömungselement wird automatisch ermittelt und es wird geprüft ob der dazwischenliegende Knoten mit einem weiteren Strömungselement verbunden ist. In diesem Fall wäre die Regelung nicht mehr anwendbar und es wird ein Fehler ausgegeben.

Der Regler `TemperatureDifferenceOfFollowingElement` kann nur für die Modelle `ControlledValve` und `ControlledPump` verwendet werden.

### 18.8.3. Regler: TemperatureDifferenceOfFollowingElement

Mit diesem Regler wird ein Ventil implementiert, welches die Temperaturdifferenz des nächsten Strömungselements (in Verknüpfungsreihenfolge) regelt. Das nächste Strömungselement wird automatisch ermittelt und es wird geprüft ob der dazwischenliegende Knoten mit einem weiteren Strömungselement verbunden ist. In diesem Fall wäre die Regelung nicht mehr anwendbar und es wird ein Fehler ausgegeben.

### 18.8.4. Regler: ThermostatValue

Der Regler `ThermostatValue` kann nur für das Modell `SimplePipe` verwendet werden.

### 18.8.5. Regler: MassFlux

Mit diesem Regler wird der Massenstrom des Strömungselements geregelt. Der Regler `MassFlux` kann nur für die Modelle `ControlledValve` und `ControlledPump` verwendet werden.

### 18.8.6. Regler: PumpOperation

Mit dem Regler `PumpOperation` kann eine `ConstantPressurePump` an oder aus geschaltet werden. Dies geschieht mit einer Hysterese in Abhängigkeit des Wärmestroms des folgenden Strömungselements. Es muss daher der Parameter `HeatLossOfFollowingElementThreshold` gesetzt sein. Ist der Wärmestrom des folgenden Elements 10 % höher als `HeatLossOfFollowingElementThreshold`, wird die Pumpe angeschaltet. Ist der Wärmestrom 10 % kleiner als `HeatLossOfFollowingElementThreshold`, wird die Pumpe ausgeschaltet. Andernfalls wird der vorherige Zustand beibehalten.

!

Dieser Regler wurde speziell implementiert um die dezentrale Umwälzpumpe in einer Wärmepumpe zu steuern.

### 18.8.7. Regler: PressureDifferenceWorstpoint

Mit dem Regler `PressureDifferenceWorstpoint` kann eine Schlechtpunktregelung mit einer `ControlledPump` implementiert werden. Die Schlechtpunktregelung wählt automatisch ein `Strömungselement` bzw. eine Gruppe von `Strömungselementen` aus welche den niedrigsten Druckverlust haben. Dieser Druckverlust wird dann nach dem Sollwert `PressureDifferenceSetpoint` geregelt. Dabei werden alle `Strömungselementen` berücksichtigt welche in dem `DataTable ObservedPressureDifferenceElementIds` angegeben wurden. Dieser `DataTable` muss in der Definition des Strömungselements der Pumpe angegeben werden. Ein Beispiel einer Implementierung zeigt [Beispiel 83](#).

!!

Bei Verwendung dieses Reglers sollte unbedingt ein PI-Regler mit niedrigem P-Anteil verwendet werden, andernfalls treten numerische Probleme auf

#### Beispiel 83. Beispiel einer Implementierung eines Schlechtpunktreglers

```
<ControlElements>
  <HydraulicNetworkControlElement id="1" controlledProperty="PressureDifferenceWorstpoint" modelType="Constant"
  controllerType="PIController" >
```

```

É <IBK:Parameter name="Kp" unit="---">0.001</IBK:Parameter>
É <IBK:Parameter name="Ki" unit="---">0.1</IBK:Parameter>
É <IBK:Parameter name="PressureDifferenceSetpoint" unit="Pa">20000</IBK:Parameter>
É </HydraulicNetworkControlElement>
</ControlElements>

...

<HydraulicNetworkElement id="301" inletNodeId="7" outletNodeId="1" componentId="2" controlId="1"
display Name="Controlled pump">
É <ObservedPressureDifferenceIds> 1: 401; 2: 501; 3: 601</ObservedPressureDifferenceIds>
</HydraulicNetworkElement>

<HydraulicNetworkElement id="401" inletNodeId="4" outletNodeId="7" componentId="3" display Name="hx1">
...
</HydraulicNetworkElement>

<HydraulicNetworkElement id="501" inletNodeId="4" outletNodeId="7" componentId="3" display Name="hx2">
...
</HydraulicNetworkElement>

<HydraulicNetworkElement id="601" inletNodeId="5" outletNodeId="7" componentId="3" display Name="hx3">
...
</HydraulicNetworkElement>

```

Der `DataTable ObservedPressureDifferenceIds` hat die Struktur `<NODE_ID>:<ELEMENT_ID1>,<ELEMENT_ID2>; <NODE_ID>:<ELEMENT_ID1>,<ELEMENT_ID2>; É;` Dabei sind die `ELEMENT_IDs` ids der `Strömungselemente` und `NODE_IDs` sind frei wählbare IDs welche z.B. einen geometrischen Punkt definieren (z.B. eine VICUS NodeId).

Im Beispiel [Beispiel 83](#) wird zu jedem Zeitpunkt der Wärmetauscher (aus hx1, hx2, hx3) ausgewählt, welcher den niedrigsten Druckverlust hat. Dieser Druckverlust wird dann auf den Sollwert `PressureDifferenceSetpoint` geregelt.

!

Als zusätzliche Ausgaben sind `PressureDifferenceAtWorstpoint` und `WorstpointNodeId` verfügbar. Dabei gibt `WorstpointNodeId` immer die `NODE_ID` aus welche zum Strömungselement mit dem niedrigstem Druckverlust gehört (entsprechend des `DataTable ObservedPressureDifferenceIds`).

### 18.8.8. Parameter für die Regler-Logik

#### Reglersollwerte

Name	Einheit	Beschreibung	Verwendung mit <code>ControlledProperty</code>
<code>TemperatureDifferenceSetpoint</code>	C	Sollwert der Temperaturdifferenz	<code>TemperatureDifference</code> , <code>TemperatureDifferenceOfFollowingElement</code>
<code>MassFluxSetpoint</code>	kg/s	Sollwert des Massenstroms	<code>MassFlux</code>
<code>HeatLossOfFollowingElementThreshold</code>	W	Grenzwert des Wärmestroms des folgenden Elements	<code>PumpOperation</code>

Name	Einheit	Beschreibung	Verwendung mit ControlledProperty
PressureDifferenceSetpoint	Pa	Sollwert des Schlechtpunktreglers	PressureDifferenceWorstpoint

#### Reglerparameter

Name	Einheit	Beschreibung	Verwendung mit ControllerType
Kp	---	Verstärkung des Reglers	PController, PIController
Ki	---	Integralwert des Reglers	PIController

## 18.9. Ausgaben

Die Ergebnisgrößen eines thermo-hydraulischen Netzwerkmodells werden wie folgt definiert. Als Referenzierungstyp dient entweder **Network** für Ausgaben des Netzwerks insgesamt, oder **NetworkElement** für die Adressierung individueller Strömungselemente (siehe [Beispiel 84](#)).

*Beispiel 84. Objektlist für die Referenzierung eines Netzwerks mit der ID 1 und ausgewählte Elemente des Netzwerks*

```
<ObjectLists>
  <ObjectList name="the Network">
    <FilterID>1</FilterID> <!-- ID of network -->
    <ReferenceType>Network</ReferenceType>
  </ObjectList>
  <ObjectList name="Pipes">
    <FilterID>1,3</FilterID> <!-- IDs of flow elements -->
    <ReferenceType>NetworkElement</ReferenceType>
  </ObjectList>
</ObjectLists>
```

### 18.9.1. Verfügbare Ausgaben

Die Ausgaben der thermo-hydraulischen Netzwerkberechnung werden je nach Referenzierungstyp in eigenen Dateien mit dem Namensschema:

¥ **network\_<griname>.tsv** (für Ausgaben mit Referenztyp **Network**)

¥ **network\_elements\_<griname>.tsv** (für Ausgaben mit Referenztyp **NetworkElement**)

angelegt. Wie bei regulären Ausgaben (siehe [Abschnitt 15.3.2](#)) wird der Suffix **\_<griname>** weggelassen, wenn es nur eine Ausgabedatei mit einem Ausgaberaaster gibt.

Für die Analyse der Netzwerke und Übergabesysteme sind sowohl die Masseströme und Temperaturen im Inneren eines Verbindungselementes, aber auch an den Verbindungsstellen zwischen zwei Elementen von Interesse. Letzterer Fall ist beispielsweise typisch für gekoppelte Erzeuger- und Verbraucherkreisläufe, wobei eine Kontrolle der Zulauf- und Rücklauftemperatur möglich sein muss.



Da die Netzwerkvisualisierungsebene keine Knoten kennt, müssen Knotentemperaturen am Ein- und Auslass des Verbindungselementes abgegriffen werden. Ein- und Auslässe sind unabhängig von der Strömungsrichtung entsprechend der Netzwerktopologie definiert.

!

Es wird bei der Topologiedefinition eines Netzwerks mittels der `HydraulicNetworkElement` Elements von einer nominalen Strömungsrichtung ausgegangen. Deshalb werden Einlass- und Auslassknoten mittels der IDs `inletNodeId` und `outletNodeId` referenziert.

!

Je nach Bedingungen im Netzwerk ist es jedoch auch möglich, dass sich die Strömungsrichtung umkehrt, und das Medium nun auf der Einströmseite eines Rohres ausströmt. Dies wirkt sich zwar im Vorzeichen des Massestroms aus, jedoch nicht in der Bezeichnung der nur von der Einbaurichtung abhängigen Ein- und Auslässe eines Strömungselements.

Möchte man alle Knotendrücke oder Knotentemperaturen erhalten, so kann man einfach von allen Strömungselementen die Drücke am Auslass erfragen. Darüber erhält man dann alle Drücke an den jeweiligen Knoten.

### 18.9.2. Ausgaben der rein hydraulischen Netzwerkberechnung

Für jedes Strömungselement kann ein Massenstrom ausgegeben werden, wobei die Strömungsrichtung immer von `inletNode` zu `outletNode` positiv definiert ist. Der Massenstrom kann über die Größe `FluidMassFlux` (in kg/s) abgefragt werden (Referenztyp `NetworkElement`).

Ebenso sind für jedes Strömungselement die Drücke am Ein- und Auslass abrufbar:

¥ `InletNodePressure` in Pa

¥ `OutletNodePressure` in Pa

### 18.9.3. Ausgaben der thermo-hydraulischen Berechnung

Jedes Strömungselement hat eine (mittlere) Temperatur, welche über die Ausgabegröße `FluidTemperature` abgefragt werden kann (Referenztyp `NetworkElement`).

!

Die mittlere Temperatur eines Strömungselements kann zur Visualisierung/Farbgebung des Elements verwendet werden.

\$

Je nach physikalischer Modellierung eines Strömungselements muss die Mitteltemperatur eines Strömungselements nicht mit der Auslasstemperatur übereinstimmen (siehe Modelldokumentation). Beispiele dafür sind Speicher oder lange verlustbehaftete Rohre.

Die Temperaturen am Ein- bzw. Auslass sind (wie die Drücke) an den physischen Positionen `inletNode` und `outletNode` definiert und können ausgegeben werden. Es sind beispielsweise folgende Ausgabevariablen für den Referenztyp `NetworkElement` definiert:

¥ `InletNodeTemperature` in C

¥ `OutletNodeTemperature` in C

¥ `FlowElementHeatLoss` in W - Wärmestrom abgegeben vom Strömungselement (Energie wird dem Fluid in diesem Element entzogen). Positive Werte bedeuten Abkühlen des Mediums (Wärmeverlust).



Die Auswahl einzelner Elemente via ID kann über Objektlisten recht flexibel erfolgen.

Je nach verwendetem Strömungselement sind individuelle zusätzliche Ausgabegrößen abrufbar (siehe Dokumentation der Strömungselemente).

#### 18.9.4. Ausgaben des Netzwerks

Es gibt seitens des Netzwerks selbst auch Variablen, welche für ein gesamtes Netzwerk abgerufen werden können (Referenztyp `Network`):

¥ `FluidMassFluxes` - Masseströme [kg/s] durch alle Strömungselemente des Netzwerks

¥ `ActiveLayerThermal Load` - Wärmestrom [W] in beheizte Schichten einer Konstruktion

¥ `NetworkZoneThermal Load` - Wärmestrom [W] in eine Zone

Diese Variablen sind vektor-wertige Größen und es muss die *Id* des jeweils angeforderten Vektorelements verwendet werden. [Beispiel 85](#) zeigt die Definition einer ID-basierten Ausgabe für Massenströme des Netzwerks.

*Beispiel 85. Beispiel für Ausgabedefinitionen mit Network als Referenztyp*

```
<!-- Outputs go to file 'network.tsv' -->
<OutputDefinition>
  <!-- We choose the flow through the second element
  (pipe 101) as reference flux for the entire network -->
  <Quantity>FluidMassFluxes[id=101]</Quantity>
  <ObjectName>Entire network</ObjectName>
  <GridName>hourly</GridName>
</OutputDefinition>
```

## 19. Functional Mock-Up Interface

In diesem Abschnitt wird die Parametrisierung für den FMU-Export/Import beschrieben. Der eigentliche Export von NANDRAD-FMUs erfolgt mit dem Tool `NandradFMUGenerator`.

Die Spezifikation der FMU Schnittstelle der zu generierenden NANDRAD-FMU wird im XML-Element `FMDescription` definiert.

*Beispiel 86. Definition einer FMU-Schnittstelle*

```
<FMDescription>
```

```

<Model Name>Ideal HeaterCoolerFixedMassFlow</Model Name>
<InputVariables>
  <FMIVariableDefinition fmiVarName="NetworkElement(3).SupplyTemperatureSchedule" unit="K" fmiValueRef="43">
    <FmiVarDescription>Schedule parameter: SupplyTemperatureSchedule</FmiVarDescription>
    <FmiStartValue>0</FmiStartValue>
    <VarName>NetworkElement.SupplyTemperatureSchedule</VarName>
    <ObjectId>3</ObjectId>
  </FMIVariableDefinition>
</InputVariables>
<OutputVariables>
  <FMIVariableDefinition fmiVarName="Location(0).Temperature" unit="K" fmiValueRef="44">
    <FmiVarDescription>Outside temperature.</FmiVarDescription>
    <FmiStartValue>293.15</FmiStartValue>
    <VarName>Location.Temperature</VarName>
    <ObjectId>0</ObjectId>
  </FMIVariableDefinition>
  <FMIVariableDefinition fmiVarName="Zone(1).AirTemperature" unit="K" fmiValueRef="45">
    <FmiVarDescription>Room air temperature.</FmiVarDescription>
    <FmiStartValue>293.15</FmiStartValue>
    <VarName>Zone.AirTemperature</VarName>
    <ObjectId>1</ObjectId>
  </FMIVariableDefinition>
</OutputVariables>
</FMIDescription>

```

Der FMU-Modellname wird im XML-Element **Model Name** angegeben. Es gelten die üblichen Einschränkungen für die Namensgebung (Name muss einen gültigen Dateinamen ergeben). Die resultierende FMU-Datei wird nach diesem Modellnamen benannt.

Alle, von der NANDRAD-FMU importierten Variablen werden in der Liste **InputVariables** abgelegt. Die exportierten Variablen (Ausgabeveriablen) werden in der Liste **OutputVariables** definiert.

## 19.1. FMI-Variablendefinition

Das XML-Element **FMIVariableDefinition** enthält die Informationen, welche in der **modelDescription.xml** erscheinen (entsprechend FMI Standard) und zusätzlich die Zuordnung zu den zugehörigen NANDRAD-Variablen.

## Appendix A: Einheitendefinitionen

Im gesamten NANDRAD-Solver werden Einheiten *nur* für Ein-/ Ausgabezwecke verwendet. Innerhalb der Berechnungsfunktionen werden *immer* die SI-Basiseinheiten verwendet, wodurch Probleme durch Einheitenumrechnungen vermieden werden.

Das Einheitensystem in NANDRAD verwendet die Konvention, dass maximal ein / (Schrägstrich) Teil des Einheitennamens sein darf. Alle Einheiten, die dem Schrägstrich folgen, stehen im Nenner der Einheit. Exponenten stehen nur hinter der Einheit, zum Beispiel **m2**. Punkte in Exponenten werden weggelassen, z. B. **h05** für die Quadratwurzel der Stunde. Mehrere Einheiten werden einfach ohne . oder \*-Zeichen aneinandergereiht, z. B. **kWh** oder **kg/m2s**.



Bei Einheiten wird zwischen Groß- und Kleinschreibung unterschieden! Zum Beispiel ist **Deg** korrekt, während **deg** nicht als korrekte Einheit erkannt wird.

SI-Basiseinheit	Konvertierbare Einheiten
-	
---	%, 1
---/d	%/d
1/K	
1/logcm	
1/m	1/cm
1/Pa	
1/s	1/min, 1/h
J	kJ, MJ, MWh, kWh, Wh
J/K	kJ/K
J/kg	kJ/kg
J/kgK	kJ/kgK, Ws/kgK, J/gK, Ws/gK
J/m <sup>2</sup>	kJ/m <sup>2</sup> , MJ/m <sup>2</sup> , GJ/m <sup>2</sup> , J/dm <sup>2</sup> , J/cm <sup>2</sup> , kWh/m <sup>2</sup>
J/m <sup>2</sup> s	W/m <sup>2</sup> , kW/m <sup>2</sup> , MW/m <sup>2</sup> , W/dm <sup>2</sup> , W/cm <sup>2</sup>
J/m <sup>3</sup>	Ws/m <sup>3</sup> , kJ/m <sup>3</sup> , MJ/m <sup>3</sup> , GJ/m <sup>3</sup> , J/dm <sup>3</sup> , J/cm <sup>3</sup> , kWh/m <sup>3</sup>
J/m <sup>3</sup> K	kJ/m <sup>3</sup> K
J/m <sup>3</sup> s	kJ/m <sup>3</sup> s, MJ/m <sup>3</sup> s, J/dm <sup>3</sup> s, J/cm <sup>3</sup> s, J/m <sup>3</sup> h, W/m <sup>3</sup> , kW/m <sup>3</sup> , MW/m <sup>3</sup> , W/dm <sup>3</sup> , W/cm <sup>3</sup> , W/mm <sup>3</sup>
J/mol	kJ/mol
J/s	J/h, J/d, kJ/d, W, kW, MW, Nm/s
K	C
K/m	
K/Pa	
kg	g, mg
kg/kg	g/kg, mg/kg
kg/m	g/m, g/mm, kg/mm
kg/m <sup>2</sup>	kg/dm <sup>2</sup> , g/dm <sup>2</sup> , g/cm <sup>2</sup> , mg/m <sup>2</sup>
kg/m <sup>2</sup> s	g/m <sup>2</sup> s, g/m <sup>2</sup> h, g/m <sup>2</sup> d, kg/m <sup>2</sup> h, mg/m <sup>2</sup> s, µg/m <sup>2</sup> s, mg/m <sup>2</sup> h, µg/m <sup>2</sup> h
kg/m <sup>2</sup> s <sup>05</sup>	kg/m <sup>2</sup> h <sup>05</sup>
kg/m <sup>3</sup>	kg/dm <sup>3</sup> , g/dm <sup>3</sup> , g/cm <sup>3</sup> , g/m <sup>3</sup> , mg/m <sup>3</sup> , µg/m <sup>3</sup> , log(kg/m <sup>3</sup> ), log(g/m <sup>3</sup> ), log(mg/m <sup>3</sup> ), log(µg/m <sup>3</sup> )
kg/m <sup>3</sup> s	g/m <sup>3</sup> s, g/m <sup>3</sup> h, kg/m <sup>3</sup> h, mg/m <sup>3</sup> s, µg/m <sup>3</sup> s, mg/m <sup>3</sup> h, µg/m <sup>3</sup> h
kg/m <sup>3</sup> sK	g/m <sup>3</sup> sK, g/m <sup>3</sup> hK, kg/m <sup>3</sup> hK, mg/m <sup>3</sup> sK, µg/m <sup>3</sup> sK, mg/m <sup>3</sup> hK, µg/m <sup>3</sup> hK

SI-Basiseinheit	Konvertierbare Einheiten
kg/mol	g/mol
kg/ms	
kg/s	kg/h, kg/d, g/d, g/a, mg/s, µg/s
kWh/a	
kWh/m <sup>2</sup> a	
l/m <sup>2</sup> s	l/m <sup>2</sup> h, l/m <sup>2</sup> d, mm/d, mm/h
l/m <sup>3</sup> s	l/m <sup>3</sup> h
logcm	
logm	
logPa	
Lux	kLux
m	mm, cm, dm
m/s	cm/s, cm/h, cm/d
m/s <sup>2</sup>	
m <sup>2</sup>	mm <sup>2</sup> , cm <sup>2</sup> , dm <sup>2</sup>
m <sup>2</sup> /kg	
m <sup>2</sup> /m <sup>3</sup>	
m <sup>2</sup> /s	cm <sup>2</sup> /s, m <sup>2</sup> /h, cm <sup>2</sup> /h
m <sup>2</sup> K/W	
m <sup>2</sup> s/kg	
m <sup>3</sup>	mm <sup>3</sup> , cm <sup>3</sup> , dm <sup>3</sup>
m <sup>3</sup> /m <sup>2</sup> s	m <sup>3</sup> /m <sup>2</sup> h, dm <sup>3</sup> /m <sup>2</sup> s, dm <sup>3</sup> /m <sup>2</sup> h
m <sup>3</sup> /m <sup>2</sup> sPa	m <sup>3</sup> /m <sup>2</sup> hPa
m <sup>3</sup> /m <sup>3</sup>	Vol%
m <sup>3</sup> /m <sup>3</sup> d	Vol%/d
m <sup>3</sup> /s	m <sup>3</sup> /h, dm <sup>3</sup> /s, dm <sup>3</sup> /h
m <sup>3</sup> m/m <sup>3</sup> m	m <sup>3</sup> mm/m <sup>3</sup> m
mm/m	
mol	mmol
mol/kg	mol/g
mol/m <sup>3</sup>	mol/ltr, mol/dm <sup>3</sup> , mol/cm <sup>3</sup>

SI-Basiseinheit	Konvertierbare Einheiten
Pa	hPa, kPa, Bar, PSI, Torr
Pa/m	kPa/m
Person/m2	
Rad	Deg
s	min, h, d, a, sqrt(s), sqrt(h), ms
s/m	kg/m2sPa
s/s	min/s, h/s, d/s, a/s
s2/m2	
W/K	
W/m2K	
W/m2K2	
W/m2s	W/m2h, kW/m2s, MW/m2s, W/dm2s, W/cm2s
W/mK	kW/mK
W/mK2	
W/Person	kW/Person
<i>undefined</i>	



Die Einheit **undefi ned** bedeutet *nicht initialisiert* (intern) und darf in Eingabedateien nicht verwendet werden.

## Appendix B: Mengenreferenzen

Die folgende Liste von Größen ist eine Übersicht über alle verfügbaren Ergebnisse, die als Ausgaben angefordert werden können. Welche Ausgaben tatsächlich verfügbar sind, hängt vom Projekt ab und wird in der Datei `var/output_reference_list.txt` ausgegeben (siehe Diskussion im Abschnitt [Outputs/Ergebnisse](#)).

Einige der Größen sind vektorwertige Größen, gekennzeichnet mit einem Suffix (**id,xxx**) oder (**index,xxx**). Um auf diese Werte zuzugreifen, muss die id/der Index in der Ausgabedefinition angegeben werden (siehe Erklärung und Beispiele im Abschnitt [Outputs/Ergebnisse](#)).

Referenz/Objekttyp	Menge	Einheit	Beschreibung
ConstructionIns tance	FluxHeatCondu ctionA	W	Wärmeleitungsfluss über die Schnittstelle A (in die Konstruktion).

Referenz/Objekttyp	Menge	Einheit	Beschreibung
ConstructionInstance	FluxHeatConductionB	W	Wärmeleitfluss über die Schnittstelle B (in die Konstruktion).
ConstructionInstance	LayerTemperature(index,xxx)	C	Mittlere Schichttemperatur für angeforderte Größen.
ConstructionInstance	SurfaceTemperatureA	C	Oberflächentemperatur an der Schnittstelle A.
ConstructionInstance	SurfaceTemperatureB	C	Oberflächentemperatur an Grenzfläche B.
Location	AirPressure	Pa	Luftdruck.
Location	Albedo	---	Albedo-Wert der Umgebung [0..1].
Location	AzimuthAngle	Deg	Solarer Azimut (0 - Nord).
Location	CO2- CO2Concentration	---	Umgebende CO2-Konzentration.
Location	CO2Density	kg/ m3	Ambiente CO2-Dichte.
Location	DeclinationAngle	Deg	Solare Deklination (0 - Nord).
Location	ElevationAngle	Deg	Solare Elevation (0 - am Horizont, 90 - direkt darüber).
Location	LWSkyRadiation	W/ m2	Langwellige Himmelsstrahlung.
Location	Latitude	Deg	Breitengrad.
Location	Longitude	Deg	Längengrad.
Location	MoistureDensity	kg/ m3	Feuchtedichte der Umgebung.
Location	RelativeHumidity	%	Relative Feuchte.
Location	SWRadDiffuseHorizontal	W/ m2	Diffuse kurzwellige Strahlungsflussdichte auf horizontaler Fläche.
Location	SWRadDirectNormal	W/ m2	Direkte kurzwellige Strahlungsflussdichte in normaler Richtung.
Location	Temperature	C	Außentemperatur.
Location	VaporPressure	Pa	Umgebungs-Dampfdruck.
Location	WindDirection	Deg	Windrichtung (0 - Nord).
Location	WindVelocity	m/s	Windgeschwindigkeit.

Referenz/Objekttyp	Menge	Einheit	Beschreibung
Model	VentilationHeatFlux(id,xxx)	W	Wärmestrom durch natürliche Lüftung
Model	VentilationRate(id,xxx)	1/h	Luftwechselrate (natürliche Lüftung)
Zone	AirTemperature	C	Raumlufttemperatur.
Zone	CompleteThermalLoad	W	Summe aller Wärmeströme in den Raum und Energiequellen.
Zone	ConstructionHeatConductionLoad	W	Summe der Wärmeleitungsflüsse von Konstruktionsoberflächen in den Raum.
Zone	VentilationHeatLoad	W	Wärmelast in den Raum durch natürliche Lüftung.