

NANDRAD Modell-Referenz

Andreas Nicolai <andreas.nicolai@tu-dresden.de>, Dirk Weiß, Stephan Hirth,
Katja Tribulowski, Hauke Hirsch, Anne Paepcke

Version 0.2 (10.03.2021)

Inhaltsverzeichnis

1. Überblick	1
2. NANDRAD Eingabedaten- und Projektdatei-Referenz	1
2.1. Struktur der Projektdatei	1
2.2. Grundlegende Datentypen in der NANDRAD-Projektdatei-Spezifikation	2
2.2.1. IBK:Parameter	2
2.2.2. IBK:IntPara	2
2.2.3. IBK:Flag	3
2.2.4. IBK:LinearSpline	3
2.2.5. LinearSplineParameter	3
2.3. Pfad-Platzhalter	4
2.4. Projektinformationen	5
2.5. Eingebettete Datenbanken	5
2.5.1. Materialien	5
2.5.2. Konstruktionstypen	7
2.5.3. Verglasungssysteme	8
2.6. Zonen	9
2.7. Konstruktionsinstanzen	11
2.7.1. Räumliche Diskretisierung (Finite-Volumen-Methode)	13
2.8. Interfaces (Konstruktions-Randbedingungen)	14
2.8.1. Wärmeleitung	15
2.8.2. Solare Absorption	16
2.8.3. Langwellige Emission	17
2.8.4. Dampfdiffusion	18
2.8.5. Luftstrom	19
2.9. Eingebettete Objekte (Fenster, Türen, Öffnungen...)	19
2.9.1. Fenster	20
2.10. Klimalasten	23
2.10.1. Übersicht	23
2.10.2. Spezifikation	23
2.10.3. Sonnenstrahlungsberechnung	28
2.10.4. Vorberechnete Schattierung	28
2.11. Objektlisten und Ergebnisreferenzen	30
2.11.1. Objektlisten-Definitionen	31
2.11.2. ID-Filter-Muster	31
2.12. Zeitpläne	32
2.12.1. Übersicht	32
2.12.2. Zeitplangruppen	34
2.12.3. Tagesschema-basierte Zeitpläne	34
2.12.4. Jahresschaltpläne	40

2.12.5. Variablenliste	41
2.13. Outputs/Ergebnisse	42
2.13.1. Globale Ausgabeparameter	43
2.13.2. Ausgaberraster	43
2.13.3. Ausgangsdefinitionen	45
2.13.4. Binäres Format	49
2.13.5. Solver-Logdateien	50
2.14. Globale Parameter	50
2.14.1. Simulationsparameter	51
2.14.2. Solver-Parameter	54
2.15. Modellparametrisierung	58
2.15.1. Natürliches Lüftungsmodell (Infiltration)	58
2.15.2. Steuerungsmodell für Verschattung	60
2.15.3. Internal Loads Model	61
2.16. HydraulicNetworks	62
2.16.1. Definition eines hydraulischen Netzwerks	63
2.16.2. Medieneigenschaften	65
2.16.3. Rohreigenschaften	66
2.16.4. Komponentendefinitionen	67
2.16.5. Strömungselemente	68
2.16.6. Definition der Wärmeaustauschmodells (HeatExchangeType)	70
2.16.7. Ausgaben	71
3. Referenz	74
3.1. Einheitendefinitionen	74
3.2. Mengenreferenzen	77

1. Überblick

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

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

2. NANDRAD Eingabedaten- und Projektdatei-Referenz

2.1. 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>
  </Project>
</NandradProject>
```

Mit dem optionalen **DirectoryPlaceholders** können relative Pfadplatzhalter definiert werden, die für 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
ProjectInfo	Allgemeine Projekt-Meta-Daten → Projektinformationen
Location	Klimadaten und Standorteinstellungen → Klimalasten

XML-Tag	Beschreibung
<code>SimulationParameter</code>	Allgemeine Parameter des Simulationsmodells → Simulationsparameter
<code>SolverParameter</code>	Einstellungen der numerischen Algorithmen → Solver-Parameter
<code>Zones</code>	Zonenspezifikationen → Zonen
<code>ConstructionInstances</code>	Gebäudekomponenten und Randbedingungen → Konstruktionsinstanzen
<code>HydraulicNetworks</code>	Thermohydraulische Netze → HydraulicNetworks
<code>ConstructionTypes</code>	Datenbank von mehrschichtigen Konstruktionen → Konstruktionstypen
<code>Materials</code>	Materialdatenbank → Materialien
<code>Models</code>	Modell-Parameterblöcke → Modellparametrisierung
<code>Schedules</code>	Definition von geplanten Parametern → Zeitpläne
<code>Outputs</code>	Definition von Ausgaben → Outputs/Ergebnisse
<code>ObjectLists</code>	Definition von Objektlisten/Objektreferenzgruppe → Objektlisten und Ergebnisreferenzen

2.2. Grundlegende Datentypen in der NANDRAD-Projektdatenspezifikation

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

2.2.1. 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="RelTol" 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.

2.2.2. IBK:IntPara

Wird für Flaggen 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. 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>
```

2.2.3. 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**.

2.2.4. IBK:LinearSpline

Eine linearere 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="ThermalLoad">
  <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>
```

2.2.5. LinearSplineParameter

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

Beispiel 6. LinearSplineParameter Definition

```
<LinearSplineParameter name="ThermalLoad" 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
Name	Spezifischer Name, der sich auf den Raumtyp bezieht, für den der Jahresplan gesetzt wird	string	required
interpolationMethod	Gibt die Interpolationsmethode zwischen den definierten y-Werten an. <ul style="list-style-type: none">• constant - konstante Interpolation (Werte konstant während des Zeitschritts)• linear - lineare Interpolation (Werte linear interpoliert zwischen Zeitschritten)	Schlüsselwort	required
WrapMethod	Gibt an, was getan werden soll, wenn Werte mit x-Werten außerhalb des x-Wertebereichs angefordert werden. <ul style="list-style-type: none">• continuous - konstante Extrapolation (ersten bzw. letzten Wert nehmen)• cyclic - zyklische Anpassung mit der modellspezifischen Periodenlänge anwenden (z. B. ein Jahr)	key	required

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

2.3. 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:

```
<DirectoryPlaceholders>
  <Platzhalter name="Klima DB">/home/sim/climate_DB</Placeholder>
  <Platzhalter 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.

2.4. 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.

2.5. 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.

2.5.1. Materialien

In der NANDRAD-Projektdatei beginnt der Abschnitt der Materialdatenbank mit einem XML-Tag namens `materials`.

Beispiel 8. 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
id	Eindeutige ID des Materials.	> 0	erforderlich Informations-/Fehlermeldungen verwenden).
string	<i>optional</i>	displayName	Name des Materials (wird für Informations-/Fehlermeldungen verwendet)

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
Density	kg/m3	Trockendichte des Materials.	> 0,01	erforderlich
HeatCapacity	J/kgK	Spezifische Wärmekapazität des Materials.	>= 100	erforderlich

Name	Standardeinheit	Beschreibung	Wertebereich	Verwendung
Conductivity	W/mK	Wärmeleitfähigkeit des trockenen Materials.	$\geq 1e-5$	erforderlich

2.5.2. Konstruktionstypen

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

Beispiel 9. 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
id	Eindeutige Identifikationsnummer	positive Ganzzahl (> 0)	erforderlich
displayName	Name der Konstruktion (wird für Informations-/Fehlermeldungen verwendet)	string	optional

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

Mit der Verwendung des Attributs `matId` referenzieren Schichten von Konstruktionen die verwendeten Materialien:

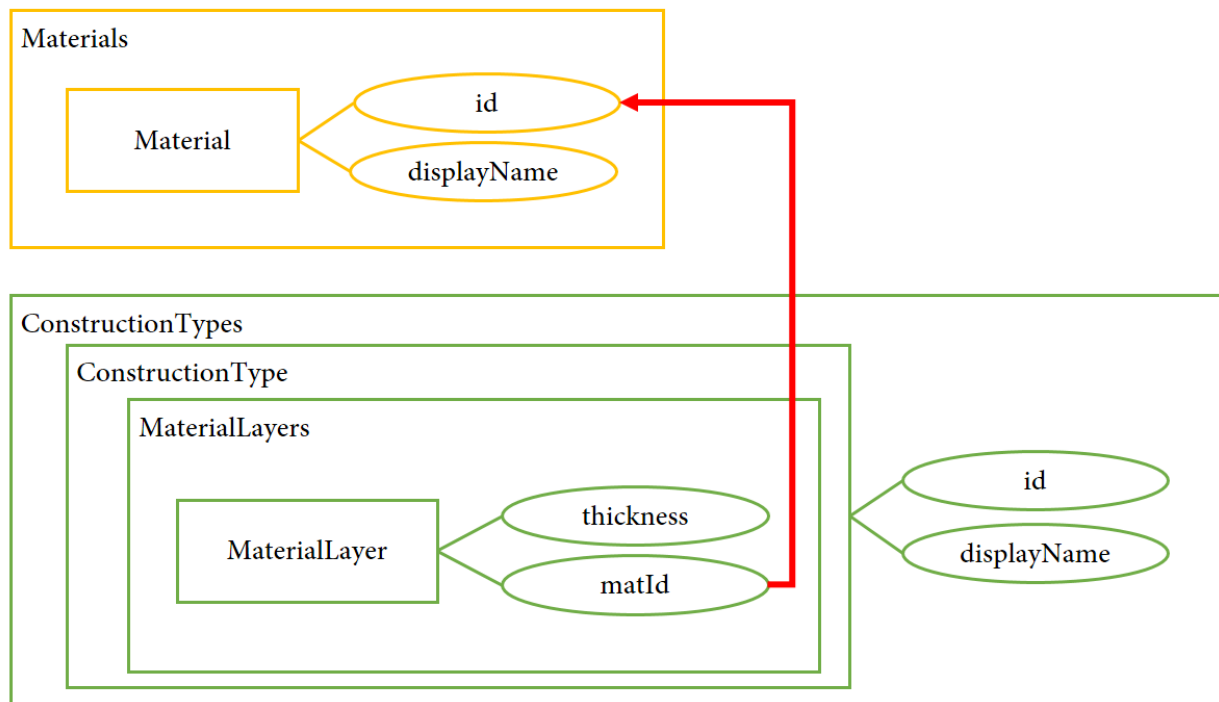


Abbildung 1. Kollaborationsdiagramm für ConstructionType- und Material-Objekte

Das **MaterialLayer** hat keine Child-Tags, da alle benötigten Daten wie oben beschrieben als XML-Attribute definiert sind.

2.5.3. Verglasungssysteme

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

Beispiel 10. Parameterdefinition für ein Verglasungssystem

```

<WindowGlazingSystems>
  <WindowGlazingSystem id="123" modelType="Standard">
    <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
id	Eindeutige Identifikationsnummer	positive integer (> 0)	<i>erforderlich</i>
displayName	Name des Verglasungssystems (wird für Informations-/Fehlermeldungen verwendet)	string	<i>optional</i>
modelType	Identifiziert die Modellkomplexität: <ul style="list-style-type: none"> • Standard - Standard-Verglasungsmodell, mit einem U-Wert (Wärmedurchgangskoeffizient) und einfallswinkelabhängigem SHGC-Wert 	string	<i>optional</i>

Skalare Parameter werden innerhalb eines XML-Tags **IBK:Parameter** definiert (siehe [IBK:Parameter](#)):

Name	Standardeinheit	Beschreibung	Wertebereich	Verwendung
ThermalTransmittance	W/m2K	Wärmedurchgangskoeffizient der Verglasung	> 0	<i>erforderlich für Modelltyp Simple</i>

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

Name	Standardeinheit	Beschreibung	Wertebereich	Verwendung
SHGC	---	Solarer Wärmegevinnkoeffizient	> 0	<i>erforderlich für Modelltyp Simple</i>

2.6. 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 Erhaltungsgröße) zu berechnen.

Beispiel 11. 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
id	eindeutige Identifikationsnummer der Zone	(> 0)	erforderlich
displayName	Anzeigenname der Zone. Wird benötigt, um die Zone im Datenmodell und in Ausgaben leichter zu finden.	string	optional
type	Legt fest, ob die Zone ausgeglichen und in das Gleichungssystem einbezogen wird. <ul style="list-style-type: none">• Constant als Zone mit konstanten/vordefinierten Temperaturen (Zeitplan)• Active als Zone, die durch einen Temperaturknoten im Raum beschrieben wird• Ground als Bodenzone (berechnet die Temperatur auf Basis des Standards)	key	erforderlich

Für *konstante* Zonen wird angenommen, dass die Temperatur vorgegeben ist, während in *Active* Zonen die Temperatur berechnet wird (d. h. in den Unbekannten des Modells enthalten ist). Eine *konstante* Zone benötigt nur den Temperaturparameter.

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

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Volume	m3	Zonenluftvolumen	> 0.0	erforderlich
Area	m2	Nettonutzfläche des Erdgeschosses (für flächenbezogene Leistungen und Lasten)	> 0.0	optional
HeatCapacity	J/K	Zusätzliche Wärmekapazität (Möbel, etc.)	>= 0.0	optional

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Temperature	C	Temperatur der Zone nur verwendet, wenn constant	-70...120	(erforderlich)
RelativeHumidity	%	Relative Luftfeuchtigkeit der Zone nur verwendet, wenn constant	0...100	(erforderlich)
CO2Concentration	g/m3	CO2-Konzentration der Zone nur verwendet, wenn constant	> 0.0	(erforderlich)

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

2.7. Konstruktionsinstanzen

Konstruktionsinstanzen repräsentieren tatsächlich gebaute eindimensionale Teile der Gebäudehülle, z.B. Wände, Böden, Decken, Dächer. Die Definition von Bauinstanzen enthält bauspezifische Parameter, die von mehreren Modellen benötigt werden.

Beispiel 12. 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
id	Identifikationsnummer der Konstruktionsinstanz	> 0	<i>erforderlich</i>
displayName	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-tag:

Tabelle 7. Construction Instance Child tags

Tag	Beschreibung
ConstructionTypeId	Referenz auf ConstructionTypeId
IBK:Parameter	Verschiedene IBK:Parameter für Konstruktionsinstanz
InterfaceA	Schnittstelle für Konstruktionsinstanz Seite A
InterfaceB	Interface für Konstruktionen

- **constructionTypeId** - eindeutige Id, die den Konstruktionstyp der Konstruktionsinstanz definiert

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
Orientation	Deg	Ausrichtung der Wand wenn eine Schnittstelle eine solare (kurzwellige) Strahlungs-Randbedingung hat, ist sie <i>erforderlich</i>	0...360	<i>erforderlich / optional</i>
Inclination	Deg	Neigung der Wand <ul style="list-style-type: none"> • 0 Deg - 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>
Area	m2	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.

2.7.1. 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 2 veranschaulicht die Wirkung verschiedener Dehnungsfaktoren

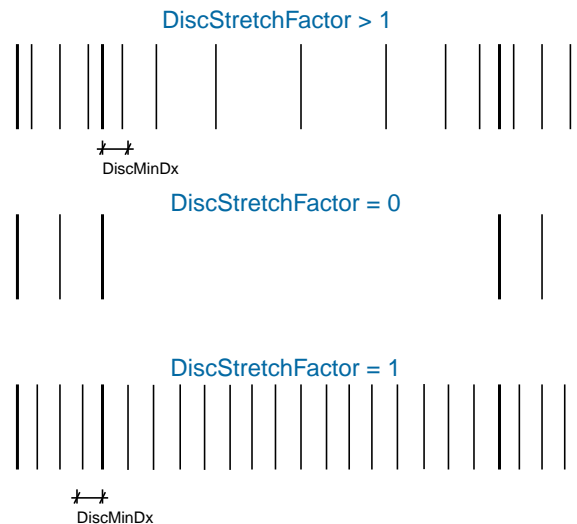


Abbildung 2. verschiedene Diskretisierungsvarianten in Abhängigkeit vom Parameter `DiscStretchFactor`

Grundsätzlich werden drei verschiedene Gittergenerierungsverfahren unterstützt:

- **minimal grid:** bei `DiscStretchFactor` = 0 erzeugt der Algorithmus ein Finites Volumen pro Materialschicht, mit Ausnahme der Randlelemente, 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.

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.

2.8. Interfaces (Konstruktions-Randbedingungen)

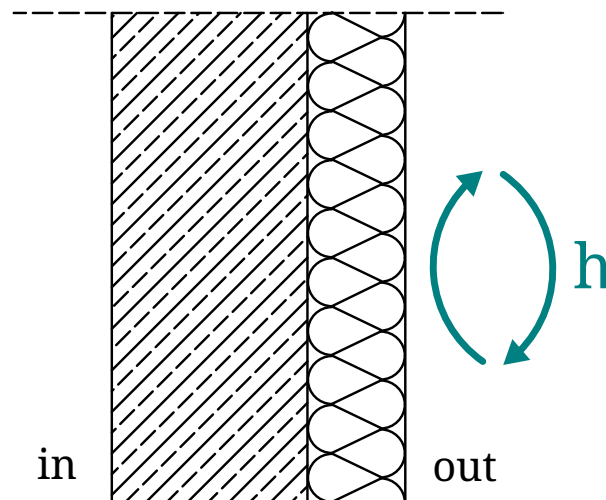
Die Interfaces definieren Randbedingungen und Parameter für die ein oder zwei Oberflächen `InterfaceA` und `InterfaceB` einer Konstruktionsinstanz. Wenn die Konstruktionsinstanz eine adiabatische Wand definiert, wird nur ein Interface benötigt. In allen anderen Fällen werden zwei Schnittstellen benötigt. 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`.

```
<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.

2.8.1. Wärmeleitung

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



Beispiel 14. 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

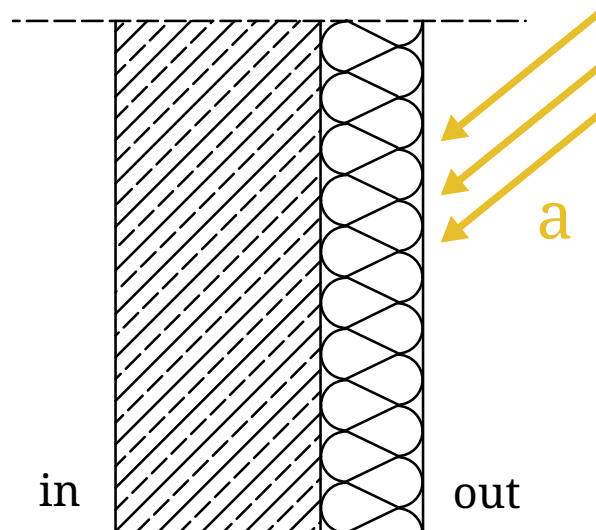
Attribut	Beschreibung	Format	Verwendung
<code>modelType</code>	Setzt den Typ des Wärmeleitungsmodells <ul style="list-style-type: none"> • <code>Constant</code> - es wird ein konstantes Modell verwendet (derzeit die einzige Option) 	key	erforderlich

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	> 0.0	erforderlich

2.8.2. 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 15. 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

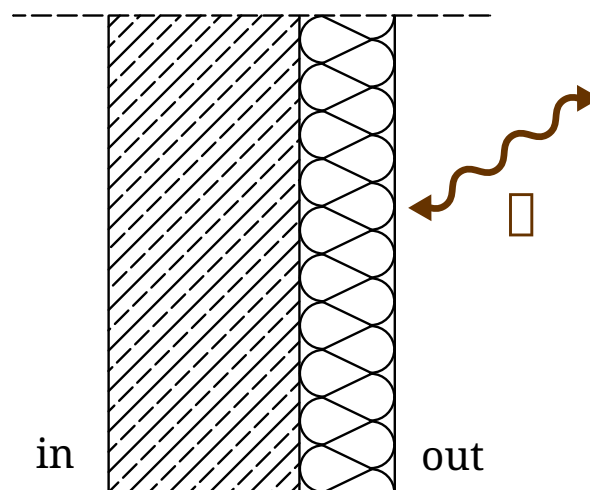
Attribut	Beschreibung	Format	Verwendung
<code>modelType</code>	Setzt den Typ des Wärmeleitungsmodells <ul style="list-style-type: none"> <code>Constant</code> - es wird ein konstantes Modell verwendet (derzeit die einzige Option) 	key	erforderlich

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

2.8.3. 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 16. parameterdefinition für 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>modellType</code>	Setzt den Typ des Wärmeleitungsmodells <ul style="list-style-type: none"> • Constant - 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>Emissivity</code>	---	Konstanter Absorptionskoeffizient	0...1	erforderlich

2.8.4. Dampfdiffusion

MUSS SPÄTER DEFINIERT WERDEN.

Die Dampfdiffusion über die Grenzfläche wird durch das XML-tag `InterfaceVaporDiffusion` beschrieben.

Beispiel 17. Parameterdefinition für Dampfdiffusion

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

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

Tabelle 11. Parameter für das InterfaceVaporDiffusion-tag

Attribut	Beschreibung	Format	Verwendung
<code>modellType</code>	Setzt den Typ des Wärmeleitungsmodells <ul style="list-style-type: none"> • Constant - 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
VaporTransferCoefficient	s/m	Dampfübergangskoeffizient	> 0.0	erforderlich

2.8.5. Luftstrom

MUSS SPÄTER DEFINIERT WERDEN.

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

Beispiel 18. 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
modelType	Setzt den Typ des Luftstroms <ul style="list-style-type: none"> Constant - 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

2.9. Eingebettete Objekte (Fenster, Türen, Öffnungen...)

Es kann mehrere Definitionen für eingebettete Objekte geben.

Beispiel 19. 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.

2.9.1. 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 20. 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 [Verglasungssysteme](#).

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 21. 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.

Fensterschattierung

Es ist möglich, vorberechnete Schattierungen sowohl auf opake als auch auf transluzente Fassadenelemente anzuwenden. Die vorberechnete Schattierung wird im Allgemeinen als globale Eigenschaft im **Location**-tag definiert (siehe [Vorberechnete Schattierung](#)).

Wenn eine vorberechnete Schattierung definiert ist, wird für **jede** opake und transluzente Fläche ein Faktor angegeben.

Wie im Abschnitt [Vorberechnete Schattierung](#) beschrieben, erfolgt die Zuordnung zwischen bereitgestellten Datenspalten und Objekt-ID über einen Identifikationsstring, der sich aus Objekttyp und ID-Nummer zusammensetzt. Zum Beispiel würde ein eingebettetes Objekt mit der ID 14 die Spaltenüberschrift/Bezeichnung *embObj.14* erhalten und eine Konstruktionsinstanz (opake Oberfläche) mit der ID 29 die Bezeichnung *conInst.29*.

Alternativ oder zusätzlich zur vorberechneten Schattierung ist es möglich eine kontrollierte Schattierung für das Fenster zu definieren.

Beispiel 22. Parameterdefinition für kontrollierte Beschattung

```
<Window glazingSystemID="123">
  ...
  <Shading modelType="Standard">
    <ControlModelID>555</ControlModelID>
    <IBK:Parameter name="ReductionFactor" unit="---">0.7</IBK:Parameter>
  </Shading>
</Window>
```

Die **Schattierung** muss mit den folgenden XML-Attributen definiert werden:

Tabelle 13. Attribute

Attribut	Beschreibung	Format	Verwendung
modelType	Setzt den Typ des Schattierungsmodells <ul style="list-style-type: none">Standard - Standardmodell, das nur einen einzigen Reduktionsfaktor und ein separates Steuerungsmodell verwendet.	key	erforderlich

Tabelle 14. Child-Tags

Element	Beschreibung	Format	Verwendung
ControlModelID	Referenz auf eine ShadingControlModel -Definition.	> 0	erforderlich für Standardmodell

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
ReductionFactor	---	Prozentualer Anteil der verbleibenden solaren Gewinne, wenn die Beschattung geschlossen ist	0...1	erforderlich für Standardmodell

Berechnung des Beschattungsfaktors auf Basis des Steuersignals

ReductionFactor = 80%

Fz in Abhängigkeit vom Steuersignal:

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

2.10. Klimalasten

2.10.1. Übersicht

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 (definiert für Außenflächen, siehe [Geometry/Constructions](#)) benötigt.

2.10.2. Spezifikation

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

Beispiel 23. Definition des Standorts

```
<Location>
  <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>
  <ClimateFileName>${Projektverzeichnis}/climate/GER_Potsdam_2017.c6b</ClimateFileName>
</Location>
```

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

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Albedo	---	Wird für die Berechnung der diffusen Sonneneinstrahlung verwendet (siehe Sonnenstrahlungsberechnung)	0...1	erforderlich
Altitude	m	wird für bestimmte höhenbezogene Parameter benötigt (TODO)	>0	optional
Longitude	Deg	Wenn angegeben, wird der Ortsparameter Longitude der Klimadatendatei überschrieben (siehe Gebäude-/Stationsstandort).	-180...180	optional

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Latitude	Deg	Wenn angegeben, wird der Ortsparameter Latitude der Klimadatendatei überschrieben (siehe Gebäude-/Stationsstandort).	-90...90	optional

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

Name	Beschreibung	Standard	Verwendung
PerezDiffuseRadiationModel	Legt fest, ob das Perez-Modell für die Berechnung der diffusen Sonnenstrahlung verwendet werden soll	false	optional

Schließlich definiert das tag **<ClimateFileName>** den Pfad zur Klimadatendatei.

Klimadateien

Derzeit werden **c6b**, **wac** und **epw** Dateien unterstützt (siehe auch Hilfe zum Werkzeug [CCM-Editor](#)).

Sie müssen den Pfad zur Klimadatendatei im **<ClimateFileName>**-tag 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

```
<ClimateFileName>GER_Potsdam_2017.c6b</ClimateFileName>
```

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 Verzeichnisplatzhalter angeben, um die Klimadatendatei *relativ* zum Speicherort der Projektdatei zu finden. 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:

```
<ClimateFileName>${Project Directory}/Klima /GER_Potsdam_2017.c6b</ClimateFileName>
```

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

Gebäude-/Stationsstandort

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

Es ist möglich, Breitengrad/Längengrad in der Projektdatei zu definieren. Wenn diese Parameter in der Projektdatei angegeben werden, müssen immer **beide** Parameter angegeben werden (und gültig sein), dann 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).

Zyklische (jährliche) und kontinuierliche (mehrjährige) Klimadaten

Die Klimadaten-Datei kann 8760 Stundenwerte für ein ganzes Jahr enthalten. Alles andere wird als beliebiger Bereich von Zeitwerten betrachtet, der ein bestimmtes Zeitintervall angibt, eventuell auch mit variierenden Zeitabständen zwischen den Datenpunkten. Die letztgenannten Klimadateien können nicht für die jährliche/zyklische Berechnung verwendet werden, sondern benötigen ein bestimmtes (passendes) Simulationszeitintervall (siehe auch Abschnitt [Simulationszeitintervall](#)).

Zyklisches Jahresklima

Die Klimadaten werden in Stundenwerten bereitgestellt. Die Interpretation dieser Werte hängt von der Art der Größe ab. NANDRAD unterscheidet zwischen Zustandsgrößen und Fluß-/Lastgrößen.

Zustandsgrößen sind:

- Temperaturen
- relative Luftfeuchten
- Luftdrücke
- Windrichtung
- Windgeschwindigkeit

Fluss-/Lastgrößen sind:

- direkte Sonnenstrahlungsintensität (in Normalrichtung der Sonne)
- diffuse solare Strahlungsintensität (in horizontaler Ebene)
- Regenlast
- langwellige Himmelsemission

Es wird erwartet, dass die Zustandsgrößen als *instantielle Werte* am *Ende jeder Stunde* überwacht werden. Substündliche Werte werden durch lineare Interpolation erhalten, wie in [Abbildung 3](#) gezeigt.

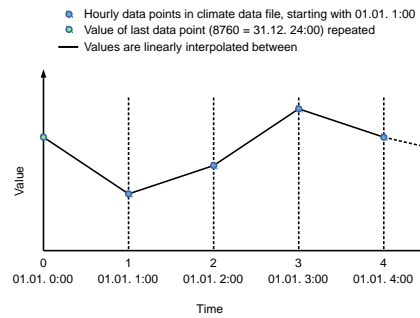


Abbildung 3. lineare Interpolation verwenden, um sub-stündliche Werte aus stündlichen Datenpunkten zu rekonstruieren

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 4](#) gezeigt.

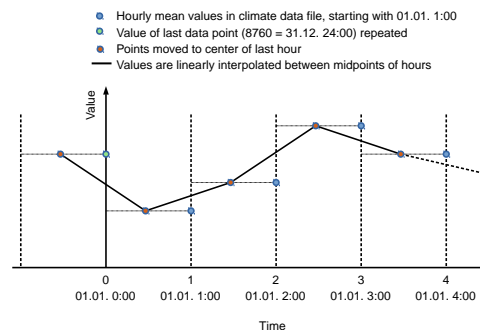


Abbildung 4. mit linearer Interpolation zur Rekonstruktion von substündlichen Werten aus stündlichen Mittelwerten

Kontinuierliche Daten

Die Klimadatendatei enthält Datenpunkte (mindestens 2), die den frühesten Start- und spätesten Endpunkt der Simulation markieren.

Wenn Sie die Simulation über die verfügbaren Klimadaten hinaus fortsetzen, werden die letzten Werte im Klimadatensatz konstant gehalten. Dies führt letztendlich 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 Stundendaten zu erzielen, müssen Klimadaten in 30-Minuten-Intervallen angegeben und interpolierte Werte am Ende und in der Mitte jeder Stunde selbst berechnet werden.

Zusätzliche Strahlungssensoren

Es ist möglich zusätzliche Ebenen (Sensoren) zu spezifizieren, um Lastausgänge für die Sonneneinstrahlung zu erzeugen. Dies geschieht durch die Angabe einer **Sensor**-Definition.

Beispiel 24. Definition eines Sensors am Standort

```
<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
id	Kennung des Sensors	> 0	erforderlich

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

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Orientation	Deg	Ausrichtung des Sensors	0...360	erforderlich
Inclination	Deg	Neigung des Sensors <ul style="list-style-type: none"> • 0 Deg - nach oben gerichtet • 90 Grad - z. B. wie eine senkrechte Wand • 180 Grad - nach unten gerichtet 	0...180	erforderlich

Einem Sensor muss eine eindeutige ID-Nummer und die obligatorischen Parameter **Orientation** und **Inclination** gegeben werden (siehe Abschnitt [Construction Instances](#) für Details zu deren Definition).

Für jeden Sensor werden 4 Ausgangsgrößen erzeugt:

- **DirectSWRadOnPlane[<sensor id>]** - direkte Sonnenstrahlungsintensität auf der Ebene in [W/m²]
- **DiffuseSWRadOnPlane[<sensor id>]** - diffuse Sonnenstrahlungsintensität auf der Ebene in [W/m²]
- **GlobalSWRadOnPlane[<sensor id>]** - globale Strahlungsintensität auf der Ebene in [W/m²] (die Summe der beiden erstgenannten Größen)
- **IncidenceAngleOnPlane[<sensor id>]** - der Einfallswinkel auf die Ebene in [Grad] (0°, wenn der Sonnenstrahl senkrecht zur Ebene steht, 90°, wenn der Strahl parallel zur Ebene steht oder wenn die Sonne unter dem Horizont ist)

Beispiel für einen Sensorausgang (siehe auch Ausgangsbeschreibung im Abschnitt [Outputs/Ergebnisse](#)).

```
<OutputDefinitionen>
...
<!-- direkte Strahlung intensiv vom Sensor mit id=2 -->
<OutputDefinition>
  <Quantity>DirektSWRadOnPlane[2]</Quantity>
  <ObjectListName>Ort</ObjectListName>
  <GridName>minütlich</GridName>
</OutputDefinition>
<!-- Einfallswinkel vom Sensor mit id=42 -->
<OutputDefinition>
  <Quantity>IncidenceAngleOnPlane[42]</Quantity>
  <ObjectListName>Ort</ObjectListName>
  <GridName>Minütlich</GridName>
</OutputDefinition>
...
</OutputDefinitions>
```

2.10.3. Sonnenstrahlungsberechnung

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

2.10.4. Vorberechnete Schattierung

Eine vorverarbeitende Software kann für jedes Flächenelement des Gebäudes den prozentualen Anteil der Fläche mit Sonneneinstrahlung berechnen. In [Abbildung 5](#) wird zum Beispiel eine Fassade teilweise verschattet.

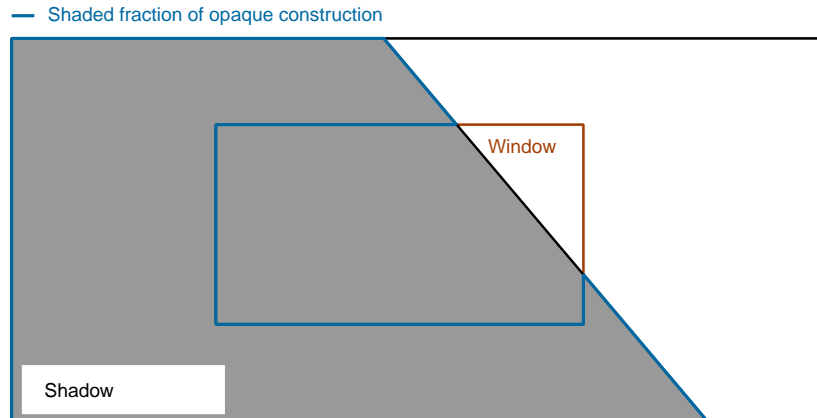


Abbildung 5. Darstellung einer teilverschatteten Fassade mit einem Fenster

Die Software kann nun den Prozentsatz der verschatteten Fläche sowohl für das opake Fassadenelement als auch für das eingebettete Fensterobjekt separat berechnen. Das Fenster ist zu ca. 80 % verschattet, und ca. 20 % der Fläche sind noch der Sonne ausgesetzt. Der Faktor, der für diese Zeit und die Fensterfläche gespeichert wird beträgt 0,2.

Der für eine Konstruktion gespeicherte Faktor beinhaltet immer auch eventuell eingebettete Objekte. [Abbildung 6](#) zeigt ein ähnliches Bild, jedoch einfacher zu berechnen.

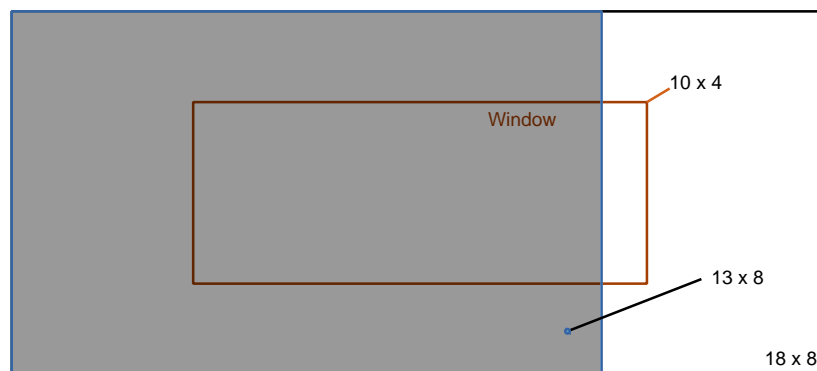


Abbildung 6. Berechnungsbeispiel für eine teilverschattete Fassade mit einem Fenster

Die Konstruktion hat eine Fläche von $18 \times 8 = 144 \text{ m}^2$. Das Fenster hat eine Fläche von $10 \times 4 = 40 \text{ m}^2$. Der Schatten auf der Konstruktion nimmt $13 \times 8 = 104 \text{ m}^2$ ein, daher ist der Schattenanteil (Faktor) für die Konstruktion: $104/144 = 72,2 \%$.

Der Schatten auf dem Fenster allein nimmt $(13 - (18 - 10)/2) \times 4 = 9 \times 4 = 36 \text{ m}^2$ ein. Der Verschattungsflächenfaktor für das Fenster allein beträgt also 90%.

Die Werte **72,2%** und **90%** werden in der Datei für die schattierten Flächenanteile gespeichert.

Die schattierte Fläche auf der opaken Fläche beträgt $104 - 36 = 68 \text{ m}^2$. Die opake Konstruktion an sich hat eine Fläche von $144 - 40 = 104 \text{ m}^2$. Der Abminderungsfaktor, der in der Gleichung für die Belastung durch die Sonneneinstrahlung verwendet werden muss beträgt also $68/104 = 65,4 \%$. Diese Berechnung wird intern durch das Solarstrahlungsmodell durchgeführt.



Denken Sie daran, den prozentualen Anteil der verschatteten Fläche für Konstruktionen **immer einschließlich aller eingebetteten Objekte** zu berechnen.

Dateiformat für vorberechnete schattierte Fraktionen

Die Datei mit den vorberechneten Schattierungsfaktoren wird im XML-tag **Location** referenziert, als Child-tag **ShadingFactorFileName**. Der hier angegebene Pfad kann ein absoluter Pfad oder ein relativer Pfad sein, der dem Platzhalter `${Project Directory}` folgt (siehe [Pfad-Platzhalter](#)).

Die Datei folgt den Regeln des **tsv**-Dateiformats (siehe *PostProc 2* Dokumentation) und hat eine einzelne Kopfzeile. Die erste Spalte ist die Zeitspalte, mit entweder absoluten Zeitstempeln oder Zeit-offsets relativ zu Mitternacht des 1. Januar des Startjahres.

Alle anderen Spalten enthalten die berechneten Bruchwerte für die schattierten Flächen, wobei jeder Spaltenkopf die jeweilige Fläche mit einer ID-Zeichenkette identifiziert.

Die ID-Zeichenkette folgt dem Muster: `<objID>.<ID-number>`. Objekt-IDs sind entweder **conInst** oder **embObj**.

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

2.11. 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 25](#) zeigt mehrere Beispiele für Definitionen der Objektliste.

```
<ObjectLists>
  <ObjectList name="All zones">
    <FilterID>*</FilterID>
    <ReferenzTyp>Zone</ReferenzTyp>
  </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>
```

2.11.1. Objektlisten-Definitionen

Alle Objektlisten werden innerhalb des übergeordneten tags **ObjectLists** definiert. Jede Objektlistendefinition 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)

2.11.2. ID-Filter-Muster

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

ID-Nummern müssen nur für Objekte mit gleichem Referenztyp 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.

2.12. Zeitpläne

2.12.1. 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**
- **AnnualSchedules**.

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 **AnnualSchedules** 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 **AnnualSchedules** definiert werden.

Beispiel 26. Definition eines Zeitplans

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

  <ScheduleGroups>
    ...
  </ScheduleGroups>
  <AnnualSchedules>
    ...
  </AnnualSchedules>
</Schedules>
```

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

- **FirstDayOfYear**
- **Holidays**
- **WeekEndDays**
- **Schedule**
- **IBK:Flag** mit dem Namen **EnableCyclicSchedules**

Tabelle 18. mögliche Definitionen von XML-tags

XML-Tags	Beschreibung	Format	Verwendung
FirstDayOfYear	<p>Der Tagestyp des 1. Januar (Offset des Wochentages des Startjahres.</p> <ul style="list-style-type: none"> • Mon - Montag (<i>Standard</i>) • Tue - Dienstag • Wed - Mittwoch • Thu - Donnerstag • Fri - Freitag • Sat - Samstag • Sun - Sonntag 	string	<i>optional</i>
Holidays	<p>Liste der Feiertage, gespeichert in einer kommagetrennten Liste von Zahlen. Jede Zahl stellt den "Tag des Jahres" dar. Schalttage werden nicht eingeschlossen.</p>	string	<i>optional</i>
WeekEndDays	<p>Wochenendtage.</p>	string	<i>optional</i>

XML-Tags	Beschreibung	Format	Verwendung
IBK:Flag	<ul style="list-style-type: none"> Name EnableCyclicSchedules <ul style="list-style-type: none"> Wenn auf true gesetzt, werden die Zeitpläne nach einem Jahr wiederholt. Wenn auf false gesetzt (gilt nur für jährliche Zeitpläne), werden diese jährlichen Zeitpläne nur einmal abgetastet. 	true (Standard) / false	optional

2.12.2. 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!

2.12.3. Tagesschema-basierte Zeitpläne

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

Beispiel 27. Definition einer Zeitplangruppe

```
<ScheduleGroups>
  <ScheduleGroup objectList="All Zones">
    <!-- AllDays constant -->
    <Schedule type="AllDays">
      ...
    </Schedule>
    <Schedule type="WeekDays">
      ...
    </Schedule>
  </ScheduleGroup>
</ScheduleGroups>
```

Beispiel 28. ObjectList-Definition, die Zonenobjekte auswählt und "All Zones" heißt

```
<ObjektListen>
  <ObjectList name="All Zones">
    <FilterID>*</FilterID>
    <ReferenzTyp>Zone</ReferenzTyp>
  </ObjectList>
</ObjektListen>
```

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
objectList	Verweise auf eine Objektliste mit dem angegebenen Namen	string	required

Beispiel 27 zeigt eine solche Definition und Beispiel 28 die entsprechende Objektliste.

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 Typs 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 29 zeigt eine solche Zeitplandefinition.

Danach überschreiben die Typen namens WeekEnd und WeekDay, falls definiert, die bereits definierten Zeitplanwerte nur für alle Wochentage oder Wochenendtage (Priorität 1). Weiterhin definieren die Wochentage namens Monday, Tuesday, ... 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 Holidays.

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 29. Zeitplandefinition mit Typ "AllDays"

```
<ScheduleGroup objectList="Zone01">
  <!-- Konstante "AllDays" -->
  <Schedule type="AllDays">
    <TagesZyklen>
      <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

Type	Priority	Description
Tuesday	2	Werte werden auf alle Dienstage des Zeitraums gesetzt
Wednesday	2	Werte werden auf alle Mittwoche des Zeitraums gesetzt
Thursday	2	Werte werden auf alle Donnerstage des Zeitraums gesetzt
Friday	2	Werte werden auf alle Freitage des Zeitraums gesetzt
Saturday	2	Werte werden auf alle Samstage des Zeitraums gesetzt
Sunday	2	Werte werden auf alle Sonntage des Zeitraums gesetzt
Holiday	3	Werte werden auf alle Feiertage des Zeitraums gesetzt, die im tag holidays angegeben sind

Beispiel 30 veranschaulicht die Verwendung verschiedener Zeitpläne zur Definition eines Wochenplans. Zunächst wird der grundlegende tägliche Zeitplan definiert. Dann werden spezielle Regeln für Dienstage und Wochenenden definiert. **Abbildung 7** veranschaulicht den resultierenden Zeitplan.

Beispiel 30. Zeitplandefinition mit verschiedenen Tagestypen

```

<Schedules>
  <WeekEndDays>Sat,Sun</WeekEndDays>
  <ScheduleGroups>
    <ScheduleGroup objectList="All zones">
      <!-- jeden Tag zwischen 8-10 -->
      <Schedule type="AllDays">
        <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>

```

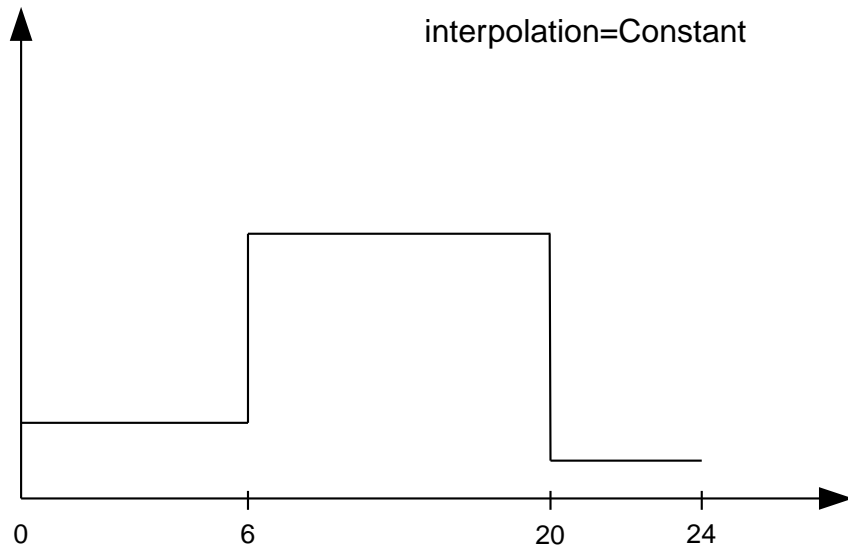


Abbildung 7. Abbildung des wöchentlichen Zeitplans, definiert durch das Beispiel [Beispiel 30](#)

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 8](#) und [Abbildung 8](#) zeigen den resultierenden Werteverlauf für die Zeitintervalle 0, 6, 20 und die entsprechenden Parameterwerte 2, 7, 1.

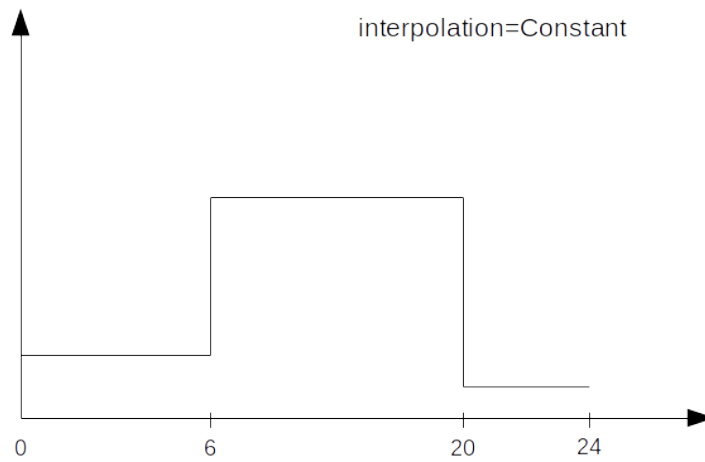


Abbildung 8. Tageszyklus mit konstantem Interpolationsmodus

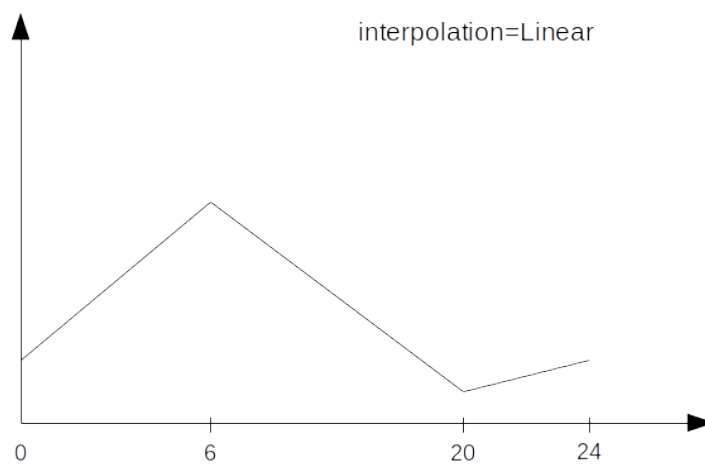


Abbildung 9. Tageszyklus mit linearem Interpolationsmodus

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 7](#) 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.

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]:val11 val12 val13; quantity2 [unit]:val21 val22 val23;...
```

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 Mengenstichwort (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 31](#) zeigt einen Tageszyklus mit zwei geplanten Mengen und drei Intervallen.

Beispiel 31. 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>
```

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 10](#) veranschaulicht die 2-minütige lineare Rampe, die direkt vor jedem neuen Intervall angewendet wird.

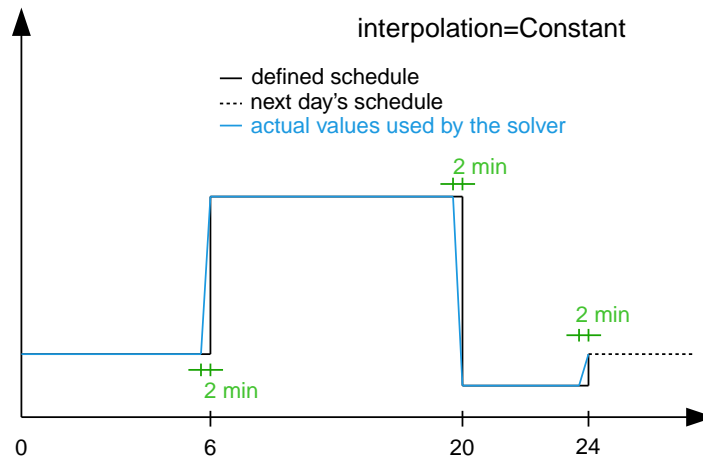


Abbildung 10. Rampen-/Schrittglättung angewandt auf Tageszyklen mit schrittweise definierten Werten



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.

2.12.4. 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.

Innerhalb des XML-tags **AnnualSchedules** gibt es ein oder mehrere XML-Tochter-tags **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 32** zeigt ein Beispiel für jährliche Zeitpläne, die innerhalb einer einzigen **ScheduleGroup** definiert sind.

Beispiel 32. Definition von Jahresplänen

```
<AnnualSchedules>
...
<ScheduleGroup objectList="All zones">
  <AnnualSchedule 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>
  </AnnualSchedule>
  <AnnualSchedule 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>
  </AnnualSchedule>
  <AnnualSchedule name="EquipmentUtilizationRatio" interpolation="linear">
    <X unit="h"> 0 2183 2184 6576 6577 8760 </X>
    <Y unit="W/Person"> 10 20 10 20 10 20 </Y>
  </AnnualSchedule>
</ScheduleGroup>
...
</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.

2.12.5. 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
ThermalLoad	W	Thermische Last (positiv oder negativ)
MoistureLoad	g/h	Feuchtelast
CoolingPower	W	Kühlleistung
LightingPower	W	Beleuchtungsleistung
DomesticWaterSetpointTemperature	C	Solltemperatur für Brauchwasser
DomesticWaterMassFlow	kg/s	Massenstrom des Brauchwasserbedarfs für die gesamte Zone (Warmwasser und Geräte)

Name	Einheit	Beschreibung
ThermalEnergyLossPerPerson	W/Person	Energie der Aktivitäten einer einzelnen Person, die nicht als Heizwärme zur Verfügung steht
TotalEnergyProductionPerPerson	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	CO2-Emissionsmassenstrom einer einzelnen Person bei einer bestimmten Tätigkeit
MassFluxRate	---	Fraktion des realen Massenstroms zum maximalen Massenstrom für verschiedene Tageszeiten
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
MaximumSolarRadiationIntensity	W/m2	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]

2.13. 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 wurde.

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

- ein Ausgabemaster, das definiert *wann* Ausgaben geschrieben werden sollen
- den Variablen-/Massennamen
- 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 eine Reihe von Log- und Datendateien,

und zwar automatisch (siehe Abschnitt [Solver-Logdateien](#)).

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

Beispiel 33. Parameter-Definition für Outputs

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

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

<Definitions>
... <!-- Tatsächliche Ausgangsdefinitionen -->
</Definitions>
</Outputs>
```

2.13.1. 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
- **IBK:Flag** - namens **BinaryFormat**: falls wahr, werden die Dateien im Binärformat geschrieben (siehe [Binäres Format](#)).

Beispiel 34. Globale Ausgabeparameter

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

2.13.2. Ausgaberraster

Ausgaberraster werden definiert, *wenn* Ausgaben geschrieben werden. Ein Ausgaberraster 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 35. Ausgaberaaster für die gesamte Simulation mit stündlichen Schritten

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

Ein Ausgaberaaster wird durch seinen Namen (obligatorisches XML-Attribut **name**) eindeutig identifiziert. Es enthält ein einzelnes untergeordnetes 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 Erklärung unten) der Wand	≥ 0.0	<i>optional</i>
End	h	die Endzeit des Intervalls (siehe Erläuterung unten)	≥ 0.0	<i>optional</i>
StepSize	h	der Abstand zwischen den Ausgängen innerhalb des Intervalls	> 0.0	<i>erforderlich</i> <i>h</i>

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.

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:

- **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 36. output Rasterauswertung

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 ist $16,5 - 12,5 = 4$ h, also ein exaktes Vielfaches von 2 h. Das Ausgaberraster ist also zu diesem Simulationszeitpunkt "aktiv" und alle Ausgänge, die mit diesem Ausgangsgitter verbunden sind, werden geschrieben.

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

Ausgaberraster 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>
```

2.13.3. Ausgangsdefinitionen

Nachfolgend finden Sie ein Beispiel für eine Ausgabedefinition:

Beispiel 37. Ausgabe der Lufttemperatur von allen Zonen in der Objektliste All zones und unter Verwendung des Ausgaberrasters hourly

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

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

XML-tag	Beschreibung	Verwendung
Quantity	Eindeutiger ID-Name der Ergebnismenge, siehe auch Mengenreferenzen	required

XML-tag	Beschreibung	Verwendung
ObjectName	Referenz auf eine Objektliste, die die Objekte identifiziert von denen Ergebnisse genommen werden sollen	<i>required</i>
GridName	Referenz auf ein Ausgabegeraster (Ausgabezeitdefinitionen)	<i>required</i>
FileName	Zielfilename	<i>optional</i>
TimeType	Methode der Zeitmittelung/Integration	<i>optional</i>

Der ID-Name der Menge 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 [object list](#) ausgewählt. Der Gittername ist der ID-Name eines [output grid](#).

Das tag **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 tag **TimeType** verwendet werden, um die zeitliche Mittelung oder die zeitliche Integration von Variablen festzulegen, siehe Abschnitt [Zeittypen](#).

Variablennamen und Variablennachschlageregeln

Mengen in Ausgabedefinitionen definieren die ID-Namen der Ausgabegrößen. Wenn ein einzelnes Element einer vektoriellen Größe angefordert wird, können diese optional einschließlich einer 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)

Ausgabedateinamen

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

Wenn kein Dateiname angegeben wird

Zielfilename(n) werden automatisch festgelegt.

Alle Ausgaben werden abhängig von der Menge gruppiert in:

- Zustände
- Ströme
- Lasten
- Sonstiges

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

- wird für Mengen vom Typ *fluxes* stattdessen die Gruppe *flux_integrals* verwendet,
- für Menge 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 (integrated) → *load_integrals_<gridname>.tsv*
- fluxes → *fluxes_<gridname>.tsv*
- fluxes (integrated) → *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!)

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 den ganzzahligen Wert

Standardmäßig (wenn der tag *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.

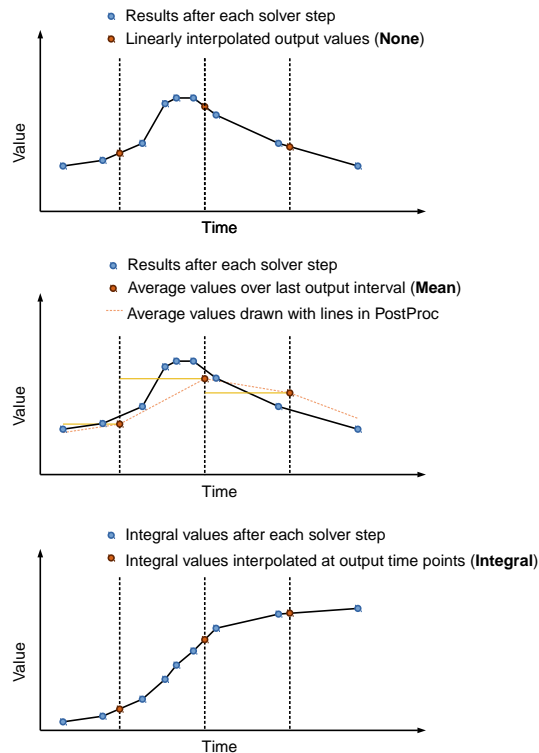


Abbildung 11. Illustration der verschiedenen **TimeType**-Optionen



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.

Beispiele

Beispiel 38. Abfrage von Oberflächentemperaturen der Konstruktionen

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

Beispiel 39. Anforderung von Energie, die der Schicht in einer Konstruktion zugeführt wird (Fußbodenheizung)

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

2.13.4. Binäres Format

Die binäre Variante von TSV-Dateien ist sehr ähnlich.

Kopfsatz:

- 64bit Ganzzahl = n (Anzahl der Spalten)
- n mal binäre Zeichenketten

Datenteil, jeder Datensatz:

Erster Datensatz: unsigned int - n (Anzahl der Spalten)

Nächste n Datensätze: Binärstrings, führende Größe (unsigned int) und Abschlusszeichen (Integritätsprüfung)

Als Nächstes ?? Datensätze: unsigned int - n (zur Überprüfung) und danach n Doubles

2.13.5. 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
    ├── output_reference_list.txt
    └── restart.bin
```

Datei	Beschreibung
<code>integrator_ccode_stats.tsv</code>	Statistik des Zeitintegrators, wird am Ende der Simulation geschrieben
<code>LES_direct_stats.tsv</code>	Statistik des Linear Equation System (LES) Solvers, wird am Ende der Simulation geschrieben
<code>progress.tsv</code>	Minimalistische Laufzeit-Fortschrittsdaten, kontinuierlich geschrieben, kann zum Verfolgen des Simulationsfortschritts vom GUI-Tool verwendet werden
<code>screenlog.txt</code>	Log-Datei für Solver-Ausgabemeldungen (wie Konsolenfensterausgaben), wird kontinuierlich geschrieben
<code>summary.txt</code>	Statistiken und Zeitangaben des Simulationslaufs, wird am Ende der Simulation geschrieben
<code>output_reference_list.txt</code>	Liste der in diesem Projekt erzeugten Größen (siehe Mengenreferenzen)
<code>restart.bin</code>	Binäre Neustartdaten (zur Fortsetzung der Integration)



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

2.14. 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.

2.14.1. Simulationparameter

Im Folgenden werden alle Simulationparameter beschrieben, siehe [Beispiel 40](#). Alle Parameter werden als `IBK:Parameter`, `IBK:Flags` oder `IBK:IntPara` gesetzt.

Beispiel 40. Simulationparameter

```
<SimulationParameter>
  <IBK:Parameter name="InitialTemperature" unit="C">5</IBK:Parameter>
  <IBK:IntPara name="DiscMaxElementsPerLayer">30</IBK:IntPara>
  <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>InitialTemperature</code>	C	Globale Anfangstemperatur für alle Objekte (<i>Zonen</i> , <i>Bauinstanzen</i> , etc)	positive double (> 0.0 K)	<i>optional</i>
<code>(*)InitialRelativeHumidity</code>	%	Globale anfängliche relative Luftfeuchtigkeit für alle Objekte, die einen Feuchtwert gesetzt haben können (<i>Zonen</i> , Luftströme in <i>Modellen</i> , etc)	0 ... 100%	<i>optional</i>

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
(*)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	<i>false</i>	<i>optional</i>
(*) EnableCO2Balance	Flag, das die Berechnung der CO2-Bilanz aktiviert, wenn aktiviert	<i>false</i>	<i>optional</i>
(*) EnableJointVentilation	Flag, das die Belüftung durch Fugen und Öffnungen aktiviert.	<i>false</i>	<i>optional</i>
(*) ExportClimateDataFMU	Flag, die den FMU-Export von Klimadaten aktiviert.	<i>false</i>	<i>optional</i>

(*) - bisher noch nicht verwendet

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.

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 **StartJahr** 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.

2.14.2. 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="RelTol" 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:Flag name="DetectMaxTimeStep">true</IBK:Flag>
  <Integrator>CVODE</Integrator>
  <LesSolver>Dense</LesSolver>
</SolverParameter>
```

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

XML-tag	Beschreibung	Verwendung
IBK:Parameter	Parameter für Fließkommazahlen	mehrfach
IBK:IntPara	Ganzzahlige Parameter	vielfach
IBK:Flag	Flags	mehrfach
Integrator	Definiert Zeitintegrator	kein/einmal
LesSolver	Definiert Solver für lineare Gleichungssysteme (LES)	kein/einmal

XML-tag	Beschreibung	Verwendung
Preconditioner	Definiert Vorkonditionierer (nur iterativer LES-Solver)	einzelne/mal

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

Name	Vorgabe Einheit	Beschreibung	Wertebereich	Vorgabe	Verwendung
RelTol	---	Relative Toleranz für die Fehlerprüfung des Solvers.	0...0.1	1E-04	<i>optional</i>
AbsTol	---	Absolute Toleranz für die Fehlerprüfung des Solvers.	0...1	1E-10	<i>optional</i>
MaxTimeStep	h	Maximal zulässiger Zeitschritt für die Integration.	positiv double (> 0.0)	1	<i>optional</i>
MinTimeStep	s	Minimal akzeptierter Zeitschritt, bevor der Solver mit einem Fehler abbricht.	positive double (> 0.0)	1E-12	<i>optional</i>
InitialTimeStep	s	Initiale Zeitschrittgröße (oder konstante Schrittgröße für ExplicitEuler-Integrator).	positive double (> 0.0)	0.1	<i>optional</i>
NonlinSolverConvCoeff	---	Koeffizient, der die Konvergenzgrenze des Solvers nichtlinearer Gleichungen reduziert. Wird von Implicit Euler nicht unterstützt.	0...1	0.1	<i>optional</i>
IterativeSolverConvCoeff	---	Koeffizientenreduzierende Konvergenzgrenze des iterativen Gleichungssolvers.	0...1	0.05	<i>optional</i>
DiscMinDx	mm	Minimale Elementbreite für Wanddiskretisierung.	positiv double (> 0.0)	2	<i>optional</i>
DiscStretchFactor	---	Stretch-Faktor für variable Wanddiskretisierungen: <ul style="list-style-type: none"> • 0 - keine Diskretisierung • 1 - äquidistant • > 1 - variabel siehe spatial discretization algorithm für Details.	positive integer (>= 0)	50	<i>optional</i>
(*)ViewfactorTileWidth	m	Maximale Abmessung einer Kachel für die Berechnung der Ansichtsfaktoren.	positive double (> 0.0)	50	<i>optional</i>

Name	Vorgabe Einheit	Beschreibung	Wertebereich	Vor- gab- e	Verwend- ung
(*) SurfaceDiscretizationDensity	---	Anzahl der Oberflächendiskretisierungselemente einer Wand in jeder Richtung.	0...1	2	<i>optional</i>
(*) ControlTemperatureTolerance	K	Temperaturtoleranz für ideales Heizen oder Kühlen.	positiv double (> 0.0)	1E-05	<i>optional</i>
(*) KinsolRelTol	---	Relative Toleranz für Kinsol-Solver.	0...1	-	<i>optional</i>
(*) KinsolAbsTol	---	Absolute Toleranz für Kinsol-Löser.	0...1	-	<i>optional</i>

(*) - bisher noch nicht verwendet

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

Name	Beschreibung	Standard	Verwend- ung
PreILUWidth	Anzahl der Nicht-Nullen in ILU	---	<i>optional</i>
MaxKrylovDim	Max. Größe der Krylov-Dimension/max. Anzahl der linearen Iterationen (nur iterative LES)	50	<i>optional</i>
MaxNonlinIter	Max. Anzahl der nicht-linearen/Newton-Iterationen	3	<i>optional</i>
MaxOrder	Max. Methodenordnung	5	<i>optional</i>
DiscMaxElementsPerLayer	Max. Anzahl der Diskretisierungselemente pro Materialschicht	20	<i>optional</i>
(*) KinsolMaxNonlinIter	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	Verwend- ung
(*) DetectMaxTimeStep	Zeitpläne prüfen, um Mindestabstände zwischen Schritten zu ermitteln und MaxTimeStep anzupassen.	<i>false</i>	<i>optional</i>
(*) KinsolDisableLineSearch	Deaktiviere Liniensuche für stationäre Zyklen.	<i>false</i>	<i>optional</i>
(*) KinsolStrictNewton	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.

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
CVODE	Wählt den CVODE -Integrator aus der Sundials-Bibliothek: implizites Mehrschrittverfahren mit fehlertestbasierter Zeitschrittanpassung und modifiziertem Newton-Raphson für nichtlineare Gleichungssysteme
ExplicitEuler	Expliziter Euler-Integrator (nur zur Fehlersuche, der Parameter InitialTimeStep bestimmt die feste Schrittweite)
ImplicitEuler	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.

Linear equation system (LES) solver

Der XML-tag **LesSolver** enthält eine Zeichenkette zur Auswahl eines bestimmten Solvers für die linearen Gleichungssysteme (**KLU** wird standardmäßig verwendet, wenn der tag fehlt).

Tabelle 23. verfügbare 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-Fähigkeiten](#) für gültige Kombinationen.

Präkonditionierer

Der XML-tag **Preconditioner** enthält eine Zeichenkette zur Auswahl eines bestimmten Preconditioners, der für iterative LES-Solver verwendet werden soll (**ILU** wird standardmäßig verwendet, wenn das tag fehlt).

Tabelle 24. verfügbare Preconditioners

Name	Beschreibung
ILU	Unvollständige LU-Faktorisierung (wenn PreILUWidth angegeben ist, wird ILU-T verwendet)

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



Eine ILU-T-Methode ist nur für `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.

Solver-Fähigkeiten

Nicht alle Integratoren und LES-Solver unterstützen alle oben genannten Optionen. Auch können nicht alle LES-Solver mit allen Integratoren kombiniert werden. Die folgende Tabelle gibt einen Überblick über die unterstützten Kombinationen und Optionen.

Tabelle 25. Fähigkeiten und unterstützte Flags/Parameter für die angebotenen Integratoren

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

2.15. Modellparametrisierung

In diesem Abschnitt werden die verschiedenen Modellparametrisierungsblöcke beschrieben.

2.15.1. Natürliches Lüftungsmodell (Infiltration)

Ein Modell für natürliche Lüftung/Infiltration definiert den Luftaustausch mit der Außenluft. Die natürliche Lüftung kann für bestimmte Zonen aktiviert werden, indem ein Modellparametersatz `NaturalVentilationModel` definiert wird. Über die Objektliste werden die Zonen mit natürlicher Lüftung ausgewählt.

```
<NaturalVentilationModel id="501" displayName="Zone vent" modelType="Constant">
  <ZoneObjectList>Office zones</ZoneObjectList>
  <IBK:Parameter name="VentilationRate" unit="1/h">0.5</IBK:Parameter>
</NaturalVentilationModel>
<NaturalVentilationModel id="502" displayName="Zone vent" modelType="Scheduled">
  <ZoneObjectList>Other zones</ZoneObjectList>
</NaturalVentilationModel>
```



Es darf nur ein Lüftungsmodellblock für eine Zone definiert werden. Es dürfen also nicht zwei Lüftungsmodellblöcke mit Objektlisten definiert werden, die beide die gleiche(n) Zone(n) enthalten.

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

Tabelle 27. Attribute

Attribut	Beschreibung	Format	Verwendung
id	Kennung des natürlichen Lüftungsmodells	> 0	erforderlich
modelType	Setzt den Typ des Lüftungsmodells <ul style="list-style-type: none"> • Konstant - Konstante Luftwechselrate • Scheduled - Luftwechselrate ändert sich nach einem definierten Zeitplan 	key	erforderlich

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

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
VentilationRate	1/h	Konstante Luftwechselrate	>= 0.0	erforderlich für konstante Modelle

Wenn der Modelltyp **Scheduled** verwendet wird, sind keine weiteren Parameter erforderlich, da die Luftwechselrate aus einem geplanten Parameter übernommen wird.

Damit das Modell funktioniert, muss der Parameter **InfiltrationRateSchedule** in einem Zeitplan definiert sein (siehe [Zeitpläne](#)).

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

Beispiel für einen Zeitplan, der den Parameter "InfiltrationRateSchedule" bereitstellt

```

<ScheduleGroup objectList="All zones">
  <!-- jeden Tag zwischen 8-10 -->
  <Schedule type="AllDays">
    <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>

```

2.15.2. 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 bestimmt. Somit kann das gleiche Regelmodell für verschiedene Verschattungsarten verwendet werden.

Beispiel 41. Parameterdefinition für Verschattungsregelungsmodell

```

<Models>
  <!-- ShadingControlModel liefert einen Wert zwischen 0 und 1
    0 = keine Reduktion (Verschattung offen)
    1 = volle Reduktion (Verschattung geschlossen)
  -->
  <ShadingControlModel id="555" displayName="Roof sensor" modelType="SingleIntensityControlled">
    <!-- Ruft die Globalstrahlung am gegebenen Sensor ab. -->
    <SensorID>21</SensorID>
    <IBK:Parameter name="MaxIntensity" unit="W/m2">200</IBK:Parameter>
    <IBK:Parameter name="MinIntensity" unit="W/m2">100</IBK:Parameter>
    <!-- Das Verschattungssteuerungsmodell liefert 1 für Werte unter Min und 0 für Werte über Max.
      Eine Umschaltung ist nur erlaubt, wenn Max/Min überschritten wurde (hysteretische Steuerung).
    -->
  </ShadingControlModel>
</Models>

```

2.15.3. Internal Loads Model

Das interne Lastenmodell 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. Es dürfen nicht zwei interne Lastmodelle existieren, die sich auf dieselben Zonen beziehen (nur eine interne Last pro Zone).

Beispiel für ein Internes Lastmodell

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

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

Tabelle 28. Attribute

Attribut	Beschreibung	Format	Verwendung
id	Kennung des natürlichen Lüftungsmodells	> 0	erforderlich
modelType	Gibt an, wie die internen Lasten angesetzt werden sollen <ul style="list-style-type: none">Konstant - Konstante Geräte-, Personen- und BeleuchtungsenergielastenScheduled - Lasten werden über Zeitplanparameter bereitgestellt.	key	erforderlich

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

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
EquipmentHeatLoadPerArea	W/m2	Komplette Gerätebelastung pro Bodenzonenfläche	>= 0.0	erforderlich für Konstantes Modell
PersonHeatLoadPerArea	W/m2	Komplette Gerätebelastung pro Bodenzonenfläche	>= 0.0	erforderlich für Konstantes Modell
LightingHeatLoadPerArea	W/m2	Komplette Gerätebelastung pro Bodenzonenfläche	>= 0.0	erforderlich für Konstantes Modell

Name	Vorgabeeinheit	Beschreibung	Wertebereich	Verwendung
EquipmentRadiationFraction	---	Prozentualer Anteil der Gerätelast, der durch Strahlung emittiert wird	≥ 0.0	erforderlich
PersonRadiationFraction	---	Prozentsatz der Gerätelast, der durch Strahlung emittiert wird	≥ 0.0	erforderlich
LightingRadiationFraction	---	Prozentsatz der Gerätelast, der durch Strahlung emittiert wird	$\geq 0,0$	erforderlich

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/m²]
- PersonHeatLoadPerAreaSchedule [W/m²]
- BeleuchtungWärmelastProFlächeRegel [W/m²]

Sowohl die geplanten als auch die konstanten internen Lastparameter sind relativ zum Nutzungsbereich, als Zonenparameter **Area** definiert. Dieser wird benötigt, sobald ein **InternalLoadsModel** definiert wurde. Daher unterscheiden sich die berechneten internen Lasten, wenn der Einsatzbereich der Zone unterschiedlich ist. Die zonenspezifischen Ausgangsgrößen heißen:

- 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 Gerätelast in Zone #3.

Die Parameter **xxxRadiationFraction** geben an, welcher Prozentsatz der berechneten internen Lasten als Strahlungsfluss flächengewichtet auf opake Oberflächen, die die Zone umschließen, aufgebracht werden soll.

2.16. HydraulicNetworks

Dieser Abschnitt beschreibt die Parametrisierung von thermo-hydraulischen Netzwerken.

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

Beispiel 42. 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 Quartierswärmenetz mit einem Gebäudeanlagennetz Wärme austauschen. Dies bedingt auch *eindeutige ID* der in *allen* Netzwerken definierten hydraulischen Netzwerkkomponenten.

2.16.1. Definition eines hydraulischen Netzwerks

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

Beispiel 43. Typische Definition eines einzelnen Netzwerks

```
<HydraulicNetwork id="1" displayName="Simple network" modelType="ThermalHydraulicNetwork"
  referenceElementId="201">
  <HydraulicFluid id="1" 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>
</HydraulicNetwork>
```

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

Tabelle 29. Attributes

Attribut	Beschreibung	Format	Verwendung
id	Eindeutige Identifizierungsnummer	> 0	<i>benötigt</i>

Attribut	Beschreibung	Format	Verwendung
<code>displayName</code>	Anzeigename/Beschreibung des Netzwerks	string	<i>optional</i>
<code>modelType</code>	Legt das Berechnungsmodell fest	string	<i>optional</i> (defaults to <code>ThermalHydraulicNetwork</code>)
<code>referenceElementId</code>	ID des Referenzelements	> 0	<i>benötigt</i>

Das optionale Attribut `modelType` beschreibt, welches Berechnungsmodell zu verwenden ist, siehe [Tabelle 30](#):

Tabelle 30. Werte für Attribut `modelType`

Attribut	Beschreibung
<code>HydraulicNetwork</code>	Nur die hydraulische Berechnung (Masseströme und Drücke) wird durchgeführt.
<code>ThermalHydraulicNetwork</code>	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.

Parameter

Je nach Modelltyp können mehrere globale Netzwerkparameter definiert werden (siehe Abschnitt [Section 2.2.1](#) für eine Beschreibung des `IBK:Parameter` Tags):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
<code>DefaultFluidTemperature</code>	C	einheitliche, konstante Fluidtemperatur zur Verwendung mit einem rein hydraulischen Modell	> 0.0	<i>benötigt für Modelltyp <code>HydraulicNetwork</code></i>
<code>InitialFluidTemperature</code>	C	Anfangsfluidtemperatur	> 0.0	<i>benötigt für Modelltyp <code>ThermalHydraulicNetwork</code></i>
<code>ReferencePressure</code>	Pa	Druck hinter dem Referenzelement	> 0.0	<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](#)

2.16.2. 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 44. Definition eines Netzwerkfluids

```
<HydraulicFluid id="1" 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 31. Attributes

Attribut	Beschreibung	Format	Verwendung
id	Eindeutige Identifikationsnummer	> 0	benötigt
displayName	Beschreibung des Fluids	string	optional



Da es nur eine Fluid-Definition per Netzwerk geben darf, ist das **id** Attribut eigentlich überflüssig. Es wird lediglich zur Aktualisierung der Programmdatenbank beim Import eines Projekts benötigt.

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

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Density	kg/m3	Dichte bei Referenztemperatur	> 0.0	benötigt
HeatCapacity	J/kgK	Spezifische Wärmekapazität	> 0.0	benötigt
Conductivity	W/mK	Wärmeleitfähigkeit bei Referenztemperatur	>= 0.0	benötigt



Die obigen Eigenschaften, insbesondere die Dichte, werden zur Vereinfachung als temperaturunabhängig konstant angenommen. Für die meisten Anwendungsfälle der thermohydraulischen 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	m2/s	Kinematische Viskosität	> 0.0	benötigt

2.16.3. Rohreigenschaften

Die Rohreigenschaften legen die physikalische/geometrischen Eigenschaften eines Rohrtyps fest. Diese werden in XML-tags wie **HydraulicNetworkPipeProperties** im Katalog **PipeProperties** mit eindeutigen IDs aufgelistet.

Beispiel 45. Definition von Rohreigenschaften

```
<PipeProperties>
  <HydraulicNetworkPipeProperties id="1">
    <IBK:Parameter name="PipeRoughness" unit="m">0.007</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 [Section 2.16.5](#)) referenziert.

Tabelle 32. 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 [Section 2.16.4.1](#)) abgemindert.

2.16.4. Komponentendefinitionen

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

Beispiel 46. 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 33. Attribute

Attribut	Beschreibung	Format	Verwendung
id	Eindeutige Identifikationsnummer der Komponente	> 0	benötigt
modelType	Modelltyp	string	benötigt

Die Parameter und Attribute sind dann abhängig vom **modelType** der Komponente und dem **modelType** des Netzwerks.

Im thermischen Modell kann für jeden Modelltyp noch ein (optionales) Wärmeaustauschmodell definiert werden (siehe [Section 2.16.6](#)).

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.

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

Modelltyp: ConstantPressurePump

Für das Model **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, > 0.0	<i>benötigt für Modelltyp ThermalHydraulicNetwork</i>
Volume	m3	Fluid volume inside the pump	> 0.0	<i>benötigt für Modelltyp ThermalHydraulicNetwork</i>

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 als Wärmequelle dem Fluid aufgeprägt.

ModellTyp: HeatExchanger

Das Model **HeatExchanger** ist ein einfacher Wärmeübertrager, welcher mit dem Fluid einen vorgegebenen Wärmestrom austauscht. Es werden diese Parameter benötigt:

Name	Einheit	Beschreibung	Wertebereich	Verwendung
HydraulicDiameter	mm	Äquivalenter hydraulischer Durchmesser (wird für die Berechnung des Strömungsquerschnitts und der Strömungsgeschwindigkeit benötigt)	> 0.0	<i>benötigt</i>
PressureLossCoefficient	---	Effektiver Druckverlustbeiwert	> 0.0	<i>benötigt</i>
Volume	m3	Fluidvolumen im Wärmetauscher	> 0.0	<i>benötigt für Modelltyp ThermalHydraulicNetwork</i>

2.16.5. 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.

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

HydraulicNetworkElement Tags haben die folgenden Attribute:

Tabelle 34. Attribute

Attribut	Beschreibung	Format	Verwendung
id	Eindeutige Identifikationsnummer des Strömungselements	> 0	benötigt
displayName	Anzeigename/Beschreibung (verwendet für Ausgaben)	string	optional
inletNodeId	ID des Einlassknotens	> 0	benötigt
outletNodeId	ID des Einlassknotens	> 0	benötigt
componentId	ID des referenzierten HydraulicNetworkComponent	> 0	benötigt
pipePropertiesId	ID des referenzierten HydraulicNetworkPipeProperties	> 0	optional (benötigt für Rohre)



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.

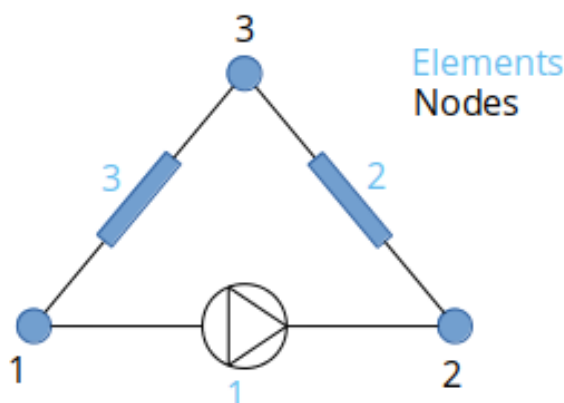


Abbildung 12. Einfaches Strömungsnetzwerk mit 3 Knoten und 3 Elementen

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 12 zeigt ein einfaches Netzwerk bestehend aus 3 Elementen. Ein solches Netzwerk würde wie folgt definiert werden (Beispiel 48).

Beispiel 48. XML-Definition eines einfachen Strömungsnetzwerks mit 3 Knoten und 3 Elementen

```
<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`.

Rohr-Elemente

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

Weiterhin **muss** für ein Rohrelement der Parameter `Length` definiert werden (siehe auch Beispiel 49):

Name	Einheit	Beschreibung	Wertebereich	Verwendung
Length	m	Rohrlänge	> 0.0	benötigt

Beispiel 49. Definition eines Rohrelements

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

2.16.6. Definition der Wärmeaustauschmodells (HeatExchangeType)

Für thermische Netzwerken kann für jedes Strömungselement (Section 2.16.5) ein Wärmeaustausch definiert werden. Dafür muss innerhalb der Definition des Strömungselements ein XML-Element `HeatExchangeType` definiert werden.

Beispiel 50. Definition von Strömungselementen mit HeatExchangeType

```
<HydraulicNetworkElement id="1" inletNodeId="1" outletNodeId="2" componentId="2" displayName="heat exchanger">
  <HydraulicNetworkHeatExchange modelType="HeatLossSpline">
    <LinearSplineParameter name="HeatLoss" interpolationMethod="linear">
      <TSVFile>${Project Directory}/climate/HeatFlux.csv?2</TSVFile>
    </LinearSplineParameter>
  </HydraulicNetworkHeatExchange>
</HydraulicNetworkElement>
<HydraulicNetworkElement id="2" inletNodeId="2" outletNodeId="3" componentId="3" pipePropertiesId="1"
displayName="pipe">
  <IBK:Parameter name="Length" unit="m">100</IBK:Parameter>
  <HydraulicNetworkHeatExchange modelType="TemperatureConstant">
    <IBK:Parameter name="ExternalHeatTransferCoefficient" unit="W/m2K">5</IBK:Parameter>
    <IBK:Parameter name="Temperature" unit="C">0</IBK:Parameter>
  </HydraulicNetworkHeatExchange>
</HydraulicNetworkElement>
```

Der **HeatExchangeType** kann folgende Werte haben:

HeatExchangeType	Beschreibung	Verwendbar für Modelltyp
TemperatureConstant	Konstante Umgebungstemperatur ist als Parameter im HydraulicNetworkElement gegeben. Es muss Zusätzlich der Parameter ExternalHeatTransferCoefficient gegeben sein.	SimplePipe, DynamicPipe
TemperatureSpline	Umgebungstemperatur ist als Zeitreihe in einer LinearSplineParameter (Section 2.2.5) gegeben. Es muss Zusätzlich der Parameter ExternalHeatTransferCoefficient gegeben sein.	SimplePipe, DynamicPipe
HeatLossConstant	Konstanter Wärmestrom (positiv aus dem Element) ist als Parameter gegeben	SimplePipe, DynamicPipe, HeatExchanger
HeatFluxSpline	Wärmestrom (positiv aus dem Element) ist als Zeitreihe in einer LinearSplineParameter (Section 2.2.5) gegeben	SimplePipe, DynamicPipe, HeatExchanger
HeatExchangeWithZoneTemperature	Temperatur wird aus Zone definiert	SimplePipe, DynamicPipe
TemperatureFMUInterface	Temperatur wird von FMU gegeben	



Wenn des XML-Element **HeatExchangeType** fehlt, wird die entsprechende Komponente als adiabatisch behandelt und verlangt entsprechend auch keine weiteren Parameter.

2.16.7. 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 51](#)).

Beispiel 51. Objektlist für die Referenzierung eines Netzwerks mit der ID 1 und ausgewählter 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>
```

Verfügbare Ausgaben

Das Netzwerk-Objekt liefert eine Vielzahl von Ergebnisgrößen für die einzelnen Strömungselemente.

Die Anforderungen an die Netzwerkausgaben richten sich allerdings nach der späteren Visualisierungsebene. Grundlegend ist davon auszugehen, dass im Postprozessing eine weitere Sicht erforderlich sein wird, welche neben dem Gebäude eine Auswertung der hydraulischen Netzwerke erlaubt. Um die Übersichtlichkeit zu wahren, wird diese Sicht von derjenigen des Gebäudes getrennt sein.

Die Netzwerkausgaben werden daher räumlich getrennt in eigenen Dateien mit dem Namensschema:

- `network_<gridname>.tsv` (für Ausgaben mit Referenztyp `Network`)
- `network_elements_<gridname>.tsv` (für Ausgaben mit Referenztyp `NetworkElement`)

angelegt. Wie bei regulären Ausgaben (siehe [Section 2.13.3.2](#)) wird der Suffix `_<gridname>` weggelassen, wenn es nur eine Ausgabedatei mit einem Ausgaberaaster gibt.

Für Analyse der Netzwerke und Übergabesysteme sind sowohl die Masseströme und Temperaturen im Innere 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 Rücklaufemperatur 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` tags 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 *geometrischen* Ein- und Auslässe eines Strömungselements.



Möchte man alle Knotendrucke oder Knotentemperaturen erhalten, so kann man einfach von allen Strömungselementen die Druck am Auslass erfragen. Darüber erhält man dann alle Drücke an den jeweiligen Knoten.

Ausgaben der hydraulischen Netzwerkberechnung

Für jedes Strömungselement kann ein Massestrom ausgehen werden, wobei die Strömungsrichtung immer von *inletNode* zu *outletNode* positiv definiert ist. Der Massestrom 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

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 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).

TODO : Anne Milestone 3

- **ZoneHeatFluxes** : Milestone 3 -

Ausgaben des Netzwerks

Zur Vereinfachung gibt es Variablen, welche für ein gesamtes Netzwerk abgerufen werden können (Referenztyp **Network**). Diese enthalten jeweils Ausgaben für alle Strömungselemente.

- **FluidMassFluxes** - Masseströme durch alle Strömungselemente des Netzwerks
- **FluidTemperatures** - Mittlere Temperaturen aller Strömungselemente des Netzwerks

Die Variablen sind vektorwertige Größen und es muss der *Index* des jeweils angeforderten Vektorelements

verwendet werden. Die Indizierung entspricht der Reihenfolge der `HydraulicNetworkElement` Tags. [Beispiel 52](#) zeigt die Definition einer indexbasierten Ausgabe für Masseströme des Netzwerks.

Beispiel 52. 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[1]</Quantity>
  <ObjectListName>Entire network</ObjectListName>
  <GridName>hourly</GridName>
</OutputDefinition>
```



Die Auswahl einzelner Elemente via ID kann über Objektlisten recht flexibel erfolgen.

Variablenamen in Ausgabedateien

Variablen für Ausgaben vom Referenztyp `NetworkElement` werden in den Ausgabedateien wie folgt angegeben: `NetworkElement(id=1).FluidMassFlux` wobei hier `id=1` die ID des ausgewählten Netzwerkelements.

3. Referenz

3.1. Einheitendefinitionen

Im gesamten NANDRAD-Solver werden Einheiten *nur* für Ein-/Ausgabebezüge 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	

SI-Basiseinheit	Konvertierbare Einheiten
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 ²	kJ/m ² , MJ/m ² , GJ/m ² , J/dm ² , J/cm ² , kWh/m ²
J/m ² s	W/m ² , kW/m ² , MW/m ² , W/dm ² , W/cm ²
J/m ³	Ws/m ³ , kJ/m ³ , MJ/m ³ , GJ/m ³ , J/dm ³ , J/cm ³ , kWh/m ³
J/m ³ K	kJ/m ³ K
J/m ³ s	kJ/m ³ s, MJ/m ³ s, J/dm ³ s, J/cm ³ s, J/m ³ h, W/m ³ , kW/m ³ , MW/m ³ , W/dm ³ , W/cm ³ , W/mm ³
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 ²	kg/dm ² , g/dm ² , g/cm ² , mg/m ²
kg/m ² s	g/m ² s, g/m ² h, g/m ² d, kg/m ² h, mg/m ² s, µg/m ² s, mg/m ² h, µg/m ² h
kg/m ² s ^{0.5}	kg/m ² h ^{0.5}
kg/m ³	kg/dm ³ , g/dm ³ , g/cm ³ , g/m ³ , mg/m ³ , µg/m ³ , log(kg/m ³), log(g/m ³), log(mg/m ³), log(µg/m ³)
kg/m ³ s	g/m ³ s, g/m ³ h, kg/m ³ h, mg/m ³ s, µg/m ³ s, mg/m ³ h, µg/m ³ h
kg/m ³ sK	g/m ³ sK, g/m ³ hK, kg/m ³ hK, mg/m ³ sK, µg/m ³ sK, mg/m ³ hK, µg/m ³ hK
kg/mol	g/mol
kg/ms	
kg/s	kg/h, kg/d, g/d, g/a, mg/s, µg/s
kWh/a	
kWh/m ² a	
l/m ² s	l/m ² h, l/m ² d, mm/d, mm/h
l/m ³ s	l/m ³ h
logcm	

SI-Basiseinheit	Konvertierbare Einheiten
logm	
logPa	
Lux	kLux
m	mm, cm, dm
m/s	cm/s, cm/h, cm/d
m/s ²	
m ²	mm ² , cm ² , dm ²
m ² /kg	
m ² /m ³	
m ² /s	cm ² /s, m ² /h, cm ² /h
m ² K/W	
m ² s/kg	
m ³	mm ³ , cm ³ , dm ³
m ³ /m ² s	m ³ /m ² h, dm ³ /m ² s, dm ³ /m ² h
m ³ /m ² sPa	m ³ /m ² hPa
m ³ /m ³	Vol%
m ³ /m ³ d	Vol%/d
m ³ /s	m ³ /h, dm ³ /s, dm ³ /h
m ³ m/m ³ m	m ³ mm/m ³ m
mm/m	
mol	mmol
mol/kg	mol/g
mol/m ³	mol/ltr, mol/dm ³ , mol/cm ³
Pa	hPa, kPa, Bar, PSI, Torr
Pa/m	kPa/m
Person/m ²	
Rad	Deg
s	min, h, d, a, sqrt(s), sqrt(h), ms
s/m	kg/m ² sPa
s/s	min/s, h/s, d/s, a/s
s ² /m ²	
W/K	
W/m ² K	
W/m ² K ²	
W/m ² s	W/m ² h, kW/m ² s, MW/m ² s, W/dm ² s, W/cm ² s

SI-Basiseinheit	Konvertierbare Einheiten
W/mK	kW/mK
W/mK2	
W/Person	kW/Person
<i>undefined</i>	



Die Einheit **undefined** bedeutet *nicht initialisiert* (intern) und darf in Eingabedateien nicht verwendet werden.

3.2. 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
ConstructionInstance	FluxHeatConductionA	W	Wärmeleitungsfluss über die Schnittstelle A (in die Konstruktion).
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	AzimutWinkel	Deg	Solarer Azimut (0 - Nord).
Location	CO2-Konzentration	---	Umgebende CO2-Konzentration.
Location	CO2Dichte	kg/m ³	Ambiente CO2-Dichte.
Location	DeclinationAngle	Deg	Solare Deklination (0 - Nord).
Location	ElevationAngle	Deg	Solare Elevation (0 - am Horizont, 90 - direkt darüber).
Location	LWSkyRadiation	W/m ²	Langwellige Himmelsstrahlung.

Referenz/Objekttyp	Menge	Einheit	Beschreibung
Location	Latitude	Deg	Breitengrad.
Location	Longitude	Deg	Längengrad.
Location	MoistureDensity	kg/m ³	Feuchtedichte der Umgebung.
Location	RelativeHumidity	%	Relative Feuchte.
Location	SWRadDiffuseHorizontal	W/m ²	Diffuse kurzwellige Strahlungsflussdichte auf horizontaler Fläche.
Location	SWRadDirectNormal	W/m ²	Direkte kurzwellige Strahlungsflussdichte in normaler Richtung.
Location	Temperatur	C	Außentemperatur.
Location	VaporPressure	Pa	Umgebungs-Dampfdruck.
Location	WindDirection	Deg	Windrichtung (0 - Nord).
Location	WindVelocity	m/s	Windgeschwindigkeit.
Model	InfiltrationHeatFlux(id,xxx)	W	Infiltrations-/Naturlüftungswärmestrom
Model	InfiltrationRate(id,xxx)	1/h	Natürliche Lüftung/Infiltration Luftwechselrate
Zone	LuftTemperatur	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	InfiltrationHeatLoad	W	Infiltration/natürliche Lüftungswärmeströme in den Raum.