EBSILON®*Professional*

Release-Notes (deutsch)

Release 16.0

steag
energy services

Release-Notes

EBSILON®*Professional*

Release 16.0

Inhaltsverzeichnis

1. Rechenkern

1.1. Neue Bauteile

1.1.1. Brennstoffzelle (Bauteil 163)

Bauteil 163 modelliert Brennstoffzellen-Stacks vom Typ SOFC (O--), PEM (H+) oder AFC (OH-). Der Detailgrad der Modellierung geht von Herstellerkurven bis zur Möglichkeit der detaillierten Modellierung der temperaturabhängigen Leitfähigkeit der einzelnen Schichten der Membran. Ausgehend von der berechneten Strom-Spannungskurve werden aus dem gesetzten Strom die zugehörige Spannung und die Energie und Massenbilanz berechnet.

1.1.2. Turbine (Kennfeld-basiert) (Bauteil 164)

Bauteil 164 modelliert eine Kennfeld-abhängige Turbine analog zum Kennfeld-abhängigen Kompressor (Bauteil 159). Dafür werden Herstellerdaten des Kennfeldes benötigt.

1.1.3. Thermischer Regenerator / Schüttgutspeicher (Bauteil 165)

Das Bauteil 165 kann zur Modellierung eines thermischen Regenerators (z. B. Winderhitzer) oder eines Schüttgutspeichers verwendet werden. Dabei wird zwischen der inneren Speichermasse und dem Gehäuse unterschieden. Das Gehäuse kann sowohl zylinderförmig als auch rechteckig sein. Die Geometrie des Gehäuses bestimmt das gesamte innere Volumen des Speichers, in dem sich die innere Speichermasse befindet.
Für weitere Details wird auf die Online-Hilfe verwiesen.

1.1.4. Latentwärmespeicher (Bauteil 166)

Das Bauteil 166 modelliert einen PCM-Speicher (Latentwärmespeicher, **PCM** steht für engl. ***phase change material***) einschließlich eines Wärmetauschers zum Be- und Entladen des Speichers. Es ist wichtig zu unterscheiden zwischen dem **PCM-Fluid** (das Medium in dem hauptsächlich die latente Wärme gespeichert wird, und das seinen Aggregatzustand - Phase - beim Be- / Entladen ändert) und dem **Fluid** (dem Wärmeträger, der durch den Wärmetauscher fließt und den Speicher be- bzw. entlädt). Im Gegensatz zum Fluid bleibt das PCM-Fluid im Speicher und fließt nicht zu oder ab. Das Fluid dagegen fließt in das Bauteil herein und aus dem Bauteil heraus. Damit wird das Bauteil 166 mit der restlichen Schaltung verbunden.
Für weitere Details wird auf die Online-Hilfe verwiesen.

1.1.5. Elektrolysezelle (Bauteil 167)

Bauteil 167 modelliert Elektrolysezellen-Stacks vom Typ SOFC (O--), PEM (H+) oder AFC (OH-). Der Detailgrad der Modellierung geht von Herstellerkurven bis zur Möglichkeit der detaillierten Modellierung der temperaturabhängigen Leitfähigkeit der einzelnen Schichten der Membran. Ausgehend von der berechneten Strom-Spannungskurve werden aus dem gesetzten Strom die zugehörige Spannung und die Energie und Massenbilanz berechnet.

1.1.6. Größenkonverter (Bauteil 168)

Dieser Bauteil dient in logischen Konstruktionen zur Abbildung von Größen, die nur als Ergebniswerte oder als abgeleitete Größen zur Verfügung stehen, auf die drei Grundgrößen Massenstrom, Druck und Enthalpie, die in Ebsilon als Variable verwendet werden. Auf diese Weise kann innerhalb der Berechnung auf alle diese Werte zugegriffen werden, beispielsweise für Regelungen oder Rechenoperationen.

Mit den Schaltern FMAPM, FMAPP und FMAPH wird eingestellt, welche Eingangsgröße auf den Massenstrom bzw. den Druck bzw. die Enthalpie der Ausgangsleitung abgebildet werden soll. Diese Abbildung erfolgt stets in den Ebsilon-Standardeinheiten: Wenn man beispielsweise eine Temperatur auf einen Massenstrom abbildet, werden 20°C auf 20 kg/s abgebildet. In den Ergebniswerten RM, RP und RH werden jedoch die ursprünglichen Dimensionen beibehalten. Dort könnte man statt 20°C sich dann auch 68°F anzeigen lassen.

Folgende Eingangsgrößen stehen zur Verfügung:

1.1.6.1. Totale und statische Größen

Solange Strömungsgeschwindigkeiten im Vergleich zur Schallgeschwindigkeit sehr klein sind, ist eine Unterscheidung zwischen totalen und statischen Größen nicht erforderlich. Bei höheren Geschwindigkeiten sind jedoch kinetische Anteile nicht mehr vernachlässigbar. Hierbei ist zu beachten, dass die auf den Leitungen in Ebsilon ausgewiesenen Größen stets Totalgrößen sind, d. h. der kinetische Anteil stets mit enthalten ist. Auf diese Weise ist es möglich, einer Leitung in Ebsilon konstante Werte – beispielsweise für die Enthalpie – zuzuordnen. Die Totalenthalpie bleibt nämlich konstant, auch wenn sich der Rohrquerschnitt und damit die Strömungsgeschwindigkeit ändert. Die Aufteilung auf statische Enthalpie und kinetische Energie hängt dagegen von der Strömungsgeschwindigkeit im jeweiligen Punkt der Leitung ab.

Die Ermittlung der statischen Größen setzt deshalb die Kenntnis der Strömungsgeschwindigkeit am jeweiligen Punkt der Leitung voraus. Diese kann deshalb in Ebsilon mit dem Bauteil 33 (Startwert) vorgegeben werden, wobei auf einer Leitung mehrere Bauteile 33 mit unterschiedlichen Geschwindigkeiten verwendet werden können. Die statischen Größen erhält man dann als Ergebniswerte des jeweiligen Startwertes.

Da die Ergebniswerte nur in EbsScripten weiterverarbeitet werden können, bietet Bauteil 168 jetzt die Möglichkeit, diese Größen auf Leitungen zu übertragen. Hierfür stehen FMAPx = 1 bis 23 zur Verfügung. Zur Erleichterung der Modellierung von Flugantrieben (der Wunsch danach war nämlich die Motivation für die Entwicklung dieses Bauteils) kann mit FMAPx = 31 auch direkt der Schub (= Massenstrom * Geschwindigkeit) übertragen werden.

1.1.6.2. Massen, Energien …

Mit FMAPx zwischen 100 und 200 können Massenströme (absolut und molar) und Energieströme (fühlbar, latent, gesamt) abgebildet werden, außerdem Heizwert, Entropie, Exergie und geodätische Höhe (der in Höhe über dem Meeresspiegel umgerechnete Luftdruck).

1.1.6.3. Stoffanteile

Mit FMAPx zwischen 200 und 300 können Stoffanteile abgebildet werden. Hierbei ist im entsprechenden FSUBSTx die Substanz anwählbar, deren Anteil abgebildet werden soll:

- 201 liefert den Massenanteil der eingestellten Substanz
- 202 liefert den Molanteil der eingestellten Substanz
- 203 liefert die Emissionsfracht, d. h. den Massenstrom der gewählten Substanz

Außerdem ist eine Analyse nach chemischen Elementen über alle Substanzen hinweg möglich, in folgenden Varianten:

- 204 liefert den Massenanteil des gewünschten Elements
- 205 liefert den Massenstrom des gewünschten Elements

Dazu muss bei FSUBSTx ein einzelnes Element ausgewählt werden, also keine Verbindung. Dabei können sowohl konventionelle Elemente als auch NASA-Elemente verwendet werden. Es ist auch eine Auswahl von „Asche" möglich. Dies beinhaltet dann den Anteil, der nicht aus den konventionellen Ebsilon-Elementen besteht.

Beispiel: eine Binärfluid-Leitung mit einem Wasser / LiBr-Gemisch kann man konventionell in H_2O und Asche aufteilen. LiBr wird als „Asche" behandelt, da es nicht aus konventionellen Ebsilon-Elementen besteht. Wenn man dagegen NASA: Li und NASA: Br verwendet, wird auch das LiBr aufgeteilt.

1.1.6.4. Phaseninformationen

301, 302 und 303 liefert den Anteil des Fluids, der sich in der gasförmigen, flüssigen bzw. festen Phase befindet.

311 bis 317 behandeln feuchte Luft:

- 311 die absolute Feuchte,
- 312 die relative Feuchte,
- 313 den Sättigungsfaktor (der für Werte < 1 mit der relativen Feuchte übereinstimmt, aber auch > 1 werden kann, wenn kondensierte Wassertröpfchen vorhanden sind),
- 314 den Partialdruck des im Fluid vorhandenen Wassers,
- 315 den Sättigungsdruck (also bei welchem Partialdruck Sättigung eintreten würde),
- 316 die Feuchtkugeltemperatur (also die Temperatur, die eine feuchte Oberfläche durch Verdunstung erreichen könnte)
- und 317 die Taupunkttemperatur (also die Temperatur, bei der sich erstmals Wassertröpfchen bilden)

321 bis 349 beziehen sich jeweils auf einen bestimmten Phasenübergang:

- 321 bis 329 auf den Phasenübergang fest / gasförmig,
- 331 bis 339 auf den Phasenübergang fest / flüssig,
- 341 bis 349 auf den Phasenübergang flüssig / gasförmig,

und zwar

- ..1 auf den Druck und ..2 auf die Temperatur des Beginns des Phasenübergangs (beim Erwärmen)
- ..3 auf den Druck und ..4 auf die Temperatur am Ende des Phasenübergangs
- ..5 die Enthalpie der Anfangsphase
- ..6 die Enthalpie der Endphase
- ..7 die Enthalpiedifferenz zwischen beiden Phasen

- ..8 die aktuelle Unterkühlung des Fluids bis zum Beginn des Phasenübergangs
- ..9 die aktuelle Überhitzung des Fluids seit dem Ende des Phasenübergangs

351 und 352 liefern Druck und Temperatur des Tripelpunktes

361 und 362 liefern Druck und Temperatur des kritischen Punktes

1.1.6.5. Stoffwertaufrufe

Mit FMAPx zwischen 400 und 500 können verschiedene Stoffwertfunktionen abgebildet werden.

1.1.6.6. Wellen und Elektroleitungen

FMAPx zwischen 500 und 600 ermöglichen den Zugriff auf Werte von Wellen und Elektroleitungen:

501 bis 507 liefen Leistungen (Wirk-/Blind-/Schein-Leistung) und mit der Leistung zusammenhängende Größen wie Leistungsfaktor (cos(phi)), Phasenverschiebung zwischen Strom und Spannung und die Anzahl der Phasen.

511 bis 512 liefern Drehzahl bzw. Frequenz und Drehmoment.

521 bis 545 ermöglichen Zugriff auf Werte zu Stromstärken (521 bis 525), Spannungen (531 bis 535) und Widerständen (542 und 545). Diese Informationen werden teilweise mit Hilfe von internen Zusatzinformationen ermitteln, die nicht Bestandteil des Gleichungssystems sind. Lediglich 521 und 531 geben den Leitungswert der Stromstärke bzw. Spannung an. FMAPx mit der Endziffer 2 liefert den Realteil, mit der Endziffer 3 den Imaginärteil der entsprechenden Größe. Mit der Endziffer 4 erhält man die Phase (bezogen auf den Nullpunkt „Masse“) und mit der Endziffer 5 den Betrag. Dieser sollte im Normalfall mit dem Leitungswert übereinstimmen. Diese Zusatzinformationen sind allerdings nicht auf jeder Elektroleitung verfügbar.

501 und 511 stehen auf Wellen und Elektroleitungen zur Verfügung, 512 nur auf Wellen, alle übrigen nur auf Elektroleitungen.

1.1.7. Biomasse Vergaser (Bauteil 169)

Bauteil 169 ist eine Erweiterung des Bauteils 96, die es erlaubt, bis zu 5 Teersubstanzen zu definieren und für die wichtigsten Eingabegrößen Kernel-Expressions zu verwenden. Die Gleichgewichtsberechnungen sind im Gegensatz zu Bauteil 96 auch druckabhängig.

1.1.8. Sammelschienenadapter (Bauteil 170)

ACHTUNG: Dieses Bauteil befindet sich noch in der Entwicklung und ist noch nicht vollständig getestet. Im Laufe der Patches zu Release 16 kommt es möglicherweise zu Änderungen im Verhalten des Bauteils (Z. B. neue/andere Spezifikationswerte, verändertes Verhalten während der Simulation, etc.)

Dieses Bauteil ist ein Adapter, der auf einer Sammelschiene zusammen mit den Bauteilen 148-150 verwendet werden kann. Hierzu werden die „äußeren“ Anschlüsse 1 und 2 in die Sammelschiene eingebaut. Zwischen den „inneren“ Anschlüssen 3 und 4 können beliebige Ebsilon-Bauteile verbaut werden.

Der Sammelschienenadapter stellt sicher, dass der Massenfluss von Anschluss 3 nach 4 IMMER POSITIV ist. D. h. je nach Strömungsrichtung in der Sammelschiene werden entweder die Leitungsdaten von Anschluss 1 -> 3 und 4 -> 2 durchgeleitet (bei positiver Richtung) bzw. 2 -> 3 und 4 -> 1 durchgeleitet (bei negativer Richtung).

Einfache Konstruktionen, die auf der Verbindung von Anschluss 3 nach 4 nicht den Massenstrom ändern, funktionieren bereits recht stabil (Z. B. Druck / Wärmesenken bzw. -quellen).
Bei Änderung des Massenstroms muss ggfs. mit Reglern gearbeitet werden.

Das Bauteil besitzt zusätzlich einen Logikanschluss (5) auf dem der Massenfluss die Werte 1 und -1 annimmt, je nachdem ob der aktuelle Massenstrom auf der Sammelschiene größer gleich 0 ist (Wert 1) oder negativ ist (Wert -1).
Diese Information kann z. B. an Anschluss 4 von Speicherbauteil 119 weitergeleitet werden, um es über die Durchströmungsrichtung zu informieren.

1.2. Erweiterungen vorhandener Bauteile

1.2.1. Rand- und Startwert (Bauteil 1, 33 und 132)

1.2.1.1. Vorgabe Lastfaktor

Es gibt jetzt einen Vorgabewert LOAD, mit dem der im Vorgabewert M eingetragene Wert multipliziert wird, um den Massenstrom auf einer Leitung zu setzen.

Dies ist hilfreich, wenn man in Unterprofilen verschiedene Lastfälle durchrechnen möchte. Man erspart sich dann die Berechnung des Massenstroms im jeweiligen Profil und braucht beim Kopieren oder Ändern der Vorgabe nur noch M im Design-Profil zu ändern.

Der Standardwert für LOAD ist 1.0, und solange man in allen Profilen LOAD auf 1.0 lässt, bleibt alles wie bisher.

1.2.2. Dampfturbine (Bauteil 6)

1.2.2.1. FOUTETAP

Da die Berechnung des polytropen Wirkungsgrades sehr zeitaufwändig ist (in manchen Fällen mit komplexen Stoffdaten hat sich dadurch die Gesamtrechenzeit des Modells vervielfacht), wird der Ergebniswert ETAP jetzt nicht mehr automatisch bei jeder Simulation berechnet. Wie schon beim Verdichter (Bauteil 24), wurde jetzt auch bei Bauteil 6 ein Schalter FOUTETAP eingeführt, mit dem man einstellen kann, ob ETAP berechnet werden soll oder nicht.

1.2.3. Wärmeaus- und -einkopplung (Bauteil 15 und 16)

1.2.3.1. Möglichkeiten zur Temperaturvorgabe

Bei diesem Bauteil musste bisher die aus- bzw. eingekoppelte Wärmeleistung auf dem Logikanschluss vorgegeben werden. Die Austrittstemperatur wurde dann aus der Wärmebilanz berechnet.

Um eine bestimmte Temperatur am Austritt einzustellen, musste bisher ein Regler verwendet werden. Dies ist nun nicht mehr notwendig, da neue Möglichkeiten zur Temperaturvorgabe implementiert wurden. Diese werden über den neuen Schalter FT ausgewählt:

FT = 0: T2 wird aus Wärmebilanz berechnet (wie bisher)
FT = 1: T2 wird durch den Vorgabewert T2SET festgelegt
FT = -1: T2 wird auf der Austrittsleitung definiert.

In den Fällen FT = 1 und FT = -1 wird die abgegebene bzw. benötigte Wärmemenge als Enthalpie auf den Logikanschluss gelegt.

Da auf dieser Logikleitung negative Werte zulässig sind, kann die Wärmeauskopplung auch als Wärmeeinkopplung verwendet werden, und umgekehrt.

1.2.4. Brennkammer (Bauteil 21)

1.2.4.1. Ergebniswert QLCO

Wenn ein Teil des Kohlenstoffs nicht vollständig zu CO_2 oxidiert wird, sondern nur zu CO, wird die im Kohlenstoff vorhandene latente Wärme nicht vollständig ausgenutzt. Dieser Verlust wird jetzt im Ergebniswert QLCO separat ausgewiesen.
Der Verlust durch CO war bisher bereits im Ergebniswert QLNB enthalten. Daran wurde nichts geändert.
Hinweis: Wieviel CO gebildet wird, wird durch die Vorgabewerte COCON bzw. ECOCON bestimmt. Der Verbrennungswirkungsgrad ETABN muss jedoch klein genug gewählt werden, um die Verluste durch CO zu ermöglichen. Andernfalls wird eine Warnung ausgegeben und der Heizwert des Rauchgases entsprechend reduziert, um die Energiebilanz einzuhalten. Diese Reduktion ist auch im Ergebniswert QLNB enthalten. Der Ergebniswert QLCO enthält jedoch den unveränderten Wert. Wenn die Warnung auftritt, muss ETABN soweit verringert werden, bis QLNB >= QLCO ist.

1.2.5. Wertanzeige (Bauteil 45)

1.2.5.1. Gefriertemperatur und -druck

Die Wertanzeige bietet jetzt analog zu FTYP = 50 (Siedetemperatur) und 51 (Siededruck) zwei neue FTYP-Varianten für den Gefrierpunkt:

- FTYP = 93 liefert zum auf der Leitung herrschenden Druck die Temperatur, bei der beim Abkühlen die Flüssigkeit zu gefrieren beginnt. Unterhalb des Tripeldrucks wird die Temperatur geliefert, bei der das Gas zu resublimieren (d. h. in die feste Phase überzugehen) beginnt.
- FTYP = 94 liefert zur auf der Leitung herrschenden Temperatur den Druck, bei der der Übergang in die feste Phase beginnt. Dies kann je nach Fluid bei steigendem oder sinkendem Druck passieren: während CO_2 sich bei steigendem Druck verfestigt, gefriert flüssiges Wasser, wenn der Druck sinkt.

1.2.5.2. Schmelztemperatur und -druck

Bei Reinstoffen wie beispielsweise Wasser sind Gefrierpunkt und Schmelzpunkt gleich. Bei Gemischen können jedoch Schmelzpunkt (Beginn des Übergangs von der festen zur flüssigen

Phase bei einer Erwärmung) und Gefrierpunkt (Beginn des Übergangs von der flüssigen in die feste Phase bei einer Abkühlung) verschieden sein. Deshalb wurden hierfür zwei weitere FTYP-Varianten implementiert:

- FTYP = 95 liefert zum auf der Leitung herrschenden Druck die Temperatur, bei der beim Erwärmen der Feststoff zu schmelzen beginnt. Unterhalb des Tripeldrucks wird die Temperatur geliefert, bei der der Feststoff zu sublimieren (d. h. in die Gasphase überzugehen) beginnt.
- FTYP = 96 liefert zur auf der Leitung herrschenden Temperatur den Druck, bei der der Übergang von der festen in die flüssige Phase beginnt.

1.2.5.3. Spezifische Gaskonstante

FTYP = 97 liefert die spezifische Gaskonstante, die durch

RS = R/MOLM

definiert ist, wobei R die universelle Gaskonstante und MOLM die Molmasse ist.

1.2.5.4. Realgasfaktor

FTYP = 98 liefert den Realgasfaktor (auch Kompressibilitätsfaktor genannt), der durch

Z = (P * V) / (RS * T) definiert ist, mit

P = Druck, V = spezifisches Volumen, RS = spezifische Gaskonstante, T = Temperatur [K]. Für ein ideales Gas ist Z = 1. Der Realgasfaktor ermöglicht deshalb eine Abschätzung, wie genau eine Berechnung als ideales Gas ist.

1.2.5.5. Normvolumenstrom nach Idealgasnäherung

Bisher wurde bei FTYP = 52 der Volumenstrom im Betriebspunkt gemäß der Formel

V/VNORM = (T/TNORM) / (P/PNORM)

auf Normbedingungen umgerechnet. Dies hat jedoch zu unerwünschten Ergebnissen geführt, wenn der Volumenstrom im Betriebspunkt nicht in Idealgasnäherung berechnet wurde, sondern beispielsweise eine Realgaskorrektur verwendet wurde.

Aus diesem Grunde erfolgt die Berechnung jetzt unabhängig von den Stoffdaten im Betriebspunkt, d. h. der Normvolumenstrom ist jetzt immer der Volumenstrom, der durch die Leitung fließen würde, wenn das Fluid ein ideales Gas bei Normbedingungen wäre:

VNORM = Ri * TNORM / PNORM,

wobei Ri die spezifische Gaskonstante ist mit $Ri = R/m_{mol}$.

R ist die universelle Gaskonstante R = 8.31441 kJ/(kmol*K) und mmol die Molmasse des Fluids.

Das Ergebnis hängt dann nicht mehr vom Betriebspunkt, sondern nur noch von der Zusammensetzung des Fluids ab.

Im Fluid vorhandene Feststoffe werden vernachlässigt. Ob Wasser in die Berechnung einbezogen werden soll oder nicht, kann wie bisher über den Schalter FNORMW eingestellt werden, ebenso wie die eventuelle Umrechnung auf einen bestimmten O_2-Gehalt.

Die Änderung bei FTYP = 52 betrifft auch das Bauteil 46 (Messwert).

1.2.5.6. Zusammensetzung der Phasen

Die Typen FTYP = 53 und FTYP = 90 (Zusammensetzung der Phasen (Massenanteile)) verhalten sich nun identisch. Der Index vor dem jeweiligen Substanznamen in der Auflistung entspricht demjenigen Index in der Liste aller verfügbaren Substanzen (siehe z. B. Liste des Vorgabewerts FSUBST).

Selbiges gilt auch für die Typen FTYP = 54 und FTYP = 91 (Zusammensetzung der Phasen (Molanteile)).

1.2.6. Abscheider (Bauteil 52)

1.2.6.1. Substanzspezifische Kernelexpression

Bei diesem Bauteil konnte die Kernelexpression EADAPT nur als gemeinsamer Korrekturfaktur für alle Abscheidegrade verwendet werden.
Jetzt gibt es eine neue Einstellung FADAPT = -2, bei dem EADAPT nicht mehr als Korrekturfaktor, sondern direkt als Abscheidegrad interpretiert wird. Werte < 0 oder > 1 werden dabei auf 0 bzw. 1 abgeschnitten.
Um für die einzelnen Substanzen unterschiedliche Abscheidegrade vorgeben zu können, wurde eine KE_Internal-Variable subst_id eingeführt, auf die man bei der Programmierung der Kernelexpression zurückgreifen kann. Das Bauteil wertet die Kermelexpression in einer Schleife für jede in der Eingangsleitung vorhandene Substanz aus und übergibt dabei in subst_id die Substanz, die gerade betrachtet wird.

Wenn man beispielsweise die Hälfte des Wassers und alles CO_2 und alles SO_2 abscheiden will, kann man folgenden Code verwenden:

```
function evalexpr:REAL;

var val:real;
        internals:array of InternalValue;
        n:integer;
        i:integer;
        subst_id:integer;

begin
        internals := keGetInternals();
        n := length( internals );
        if (n > 0) then subst_id :=internals[0].value;

        if (subst_id = 4) then // H2O
        begin
                val:=0.5;
        end
```

```
        else if (subst_id = 3 or subst_id = 5) then // CO2 und SO2
        begin
                val:=1.0;
        end
        else
        begin
                val:=0.0;
        end;
        evalexpr := val;
end;
```

1.2.6.2. Abscheidung mit bestimmter Austrittszusammensetzung

Zur Erleichterung der Modellierung chemischer Prozesse wurde mit den Modi FM = 4 und FM = 5 die Möglichkeit implementiert, das Verhältnis der Substanzen am Austritt festzulegen. Hierbei werden die Vorgabewerte J.. nicht als Abscheidegrade interpretiert, sondern als relatives Verhältnis der Substanzen untereinander, das auf der Austrittsleitung erreicht werden soll. Bei FM = 4 werden dabei die Massenanteile angegeben, bei FM = 5 die Molanteile.

Die Abscheidemenge wird durch den Vorgabewert RR (Reaktionsrate) bestimmt. Für RR = 1 wird so viel wie möglich abgeschieden, also so viel, bis von (mindestens) einer Substanz nichts mehr übrigbleibt.

1.2.7. Wärmetauscher (Bauteil 61)

1.2.7.1. Bestimmung der Wärmeübergangskoeffizienten im Auslegungsfall

Für die Auslegung eines Wärmetauschers werden in Ebsilon wahlweise bestimmte Temperaturen oder Temperaturdifferenzen (FSPECD = 1 bis 5) oder die Effektivität EFFN (FSPECD = 0) oder aber auch die Wärmetauscherfläche AN (FSPECD = 9) vorgegeben. Ebsilon ermittelt daraus bei der Auslegungsrechnung die Kenngröße KA, also das Produkt aus dem Wärmedurchgangskoeffizienten k und der Fläche A.

Der Wärmedurchgangskoeffizient wurde dabei bisher stets aus den beiden Wärmeübergangskoeffizienten AL12N und AL34N für die kalte bzw. warme Seite berechnet, die vom Anwender vorgegeben werden mussten. Bei FSPECD = 0 bis 5 hat eine Ungenauigkeit bei der Vorgabe von AL12N und AL34N im Auslegungsfall keine Auswirkungen auf den Wärmeaustausch, da dieser nur vom Produkt k*A abhängt. Um allerdings aus k*A die Wärmetauscherfläche AN zu bestimmen und auch für das Teillastverhalten ist eine genaue Kenntnis von k jedoch erforderlich, bei FSPECD = 9 auch im Auslegungsfall.

Release 16 bietet die Möglichkeit an, die Wärmeübergangskoeffizienten von Ebsilon ermitteln zu lassen. Dies geschieht mit dem neuen Schalter FAA:

- FAA = 0 ermittelt AL12N und AL34N und auch die zugehörigen Teillastexponenten EX12 und EX34 anhand der an das Bauteil angeschlossenen Fluide:
    - Ist die entsprechende Fluid-Kombination in der Datenbank enthalten, werden die Daten dieser Kombination verwendet.
    - Ist die entsprechende Fluid-Kombination nicht in der Datenbank enthalten, aber beide Fluide, werden für jedes Fluid Durchschnittswerte gebildet, die dieses Fluid in den vorhandenen Kombinationen aufweist. Der Anwender wird in einem Kommentar darauf hingewiesen.

- Ist nur eines der beiden Fluide in der Datenbank verfügbar, werden für dieses Fluid die Datenbankwerte und für das andere Fluid die Vorgabewerte verwendet. Der Anwender wird in einem Kommentar darauf hingewiesen.
- Ist keines der beiden Fluide in der Datenbank verfügbar, werden für beide Fluide die Vorgabewerte verwendet. In diesem Fall wird eine Warnung abgesetzt, da der Modus FAA = 0 in diesem Fall keinen Auswirkung hat.

Für neu eingefügte Wärmetauscher ist dieser Modus die Standardeinstellung.

- FAA = 1 verwendet die Vorgabewerte AL12N, AL34N, EX12 und EX34. Dies entspricht dem bisherigen Verhalten.
  Deshalb werden Wärmetauscher, die mit einer älteren Ebsilon-Version erstellt wurden, automatisch auf diesen Modus eingestellt, damit die Ergebnisse unverändert bleiben.
- FAA = 2 kann verwendet werden, um den Wärmeübergangskoeffizienten AL12N zu berechnen, sofern die Fläche AN bekannt ist. Diese Option steht allerdings nur bei FSPECD = 0 bis 5 zur Verfügung, da zusätzlich zur Fläche noch eine weitere Vorgabegröße benötigt wird. AL34N und beide Exponenten werden nach Möglichkeit aus der Datenbank genommen.
- FAA = 3 kann verwendet werden, um den Wärmeübergangskoeffizienten AL34N zu berechnen, sofern die Fläche AN bekannt ist. Diese Option steht allerdings nur bei FSPECD = 0 bis 5 zur Verfügung, da zusätzlich zur Fläche noch eine weitere Vorgabegröße benötigt wird. AL12N und beide Exponenten werden nach Möglichkeit aus der Datenbank genommen.
- FAA = 4 kann verwendet werden, um den Wärmeübergangskoeffizienten AL12N zu berechnen, sofern die Fläche AN bekannt ist. Diese Option steht allerdings nur bei FSPECD = 0 bis 5 zur Verfügung, da zusätzlich zur Fläche noch eine weitere Vorgabegröße benötigt wird. Anders als bei FAA = 2 werden bei FAA = 4 für AL34N und die beiden Exponenten die Vorgabewerte verwendet.
- FAA = 5 kann verwendet werden, um den Wärmeübergangskoeffizienten AL34N zu berechnen, sofern die Fläche AN bekannt ist. Diese Option steht allerdings nur bei FSPECD = 0 bis 5 zur Verfügung, da zusätzlich zur Fläche noch eine weitere Vorgabegröße benötigt wird. Anders als bei FAA = 3 werden bei FAA = 5 für AL12N und die beiden Exponenten die Vorgabewerte verwendet.

1.2.7.2. Erweiterung um transiente Berechnung

Das Bauteil 61 kann ab Release 16 auch in transienten Berechnungen verwendet werden. Eine transiente Berechnung mit dem Bauteil 61 ist allerdings nur in Off-Design-Modus möglich.

Bei der Parametrierung des Bauteils kann man zwischen 2 Ansätzen wählen:

- Der vereinfachte Ansatz mit der Vorgabe des Verhältnisses der Wärmeaustauschfläche zu der Masse der Rohre (Vorgabewert ATM) und des Verhältnisses der Wärmeaustauschfläche zum inneren Volumen der Rohre (Vorgabewert ATV)
- Der genauere Ansatz mit der Spezifikation der Rohr- und ggf. der Rippenparameter

In beiden Fällen wird die in der stationären Simulation ermittelte Fläche (AN) benutzt, um die thermische Masse der Rohre und das Rohrvolumen für die transiente Berechnung zu bestimmen.

Das Bauteil 61 benutzt das kombinierte physikalische und numerische Berechnungsmodell des Bauteils 126. Ein Schalter FINST wird benutzt, um zwischen der stationären (wie bisher) und der transienten Berechnung umzuschalten. Für weitere Details wird auf die EBSILON Online Hilfe verwiesen.

1.2.8. Gasturbine (Bauteil 106)

- Zusätzliche Ports (11, 12) für Wasser, die es erlauben, verschiedene Methoden zur Leistungssteigerung und NOx-Reduktion durch Wasserzuführung getrennt zu modellieren
- Zusätzlicher Abwärmeport (13) für eine weitere Abwärmequelle
- Erweiterung der Datenstruktur um den maximalen H_2 Gehalt (Vol%) und die Möglichkeit, die Liste der Datensätze danach zu filtern

Neue Ergebnisvariablen:

- RALAM: Luftverhältnis
- RGTCSET: Verwendeter Datensatz
- RQCOOL2: Abwärmeleistung der 2. Abwärmequelle.

1.2.9. Indirekter Speicher (Bauteil 119)

ACHTUNG: Die hier beschriebene Erweiterung zu Bauteil 119 befindet sich noch in der Entwicklung und ist noch nicht vollständig getestet. Im Laufe der Patches zu Release 16 kommt es möglicherweise zu Änderungen im Verhalten der hier beschriebenen Erweiterung.

Bei Bauteil 119 gibt es beim Spezifikationswert „FDIR“ folgende neue Werte zur Auswahl:

2: vorwärtsgerichteter Fluss
3: rückwärtsgerichteter Fluss
4: verwende Massenfluss an Anschluss 4

Zusätzlich gilt es auch Logikanschluss (4), um bei FDIR = 4 die Richtung anzuzeigen (Massenstrom größer gleich 0, dann vorwärtsgerichtet, bei negativen Massenstrom rückwärtsgerichtet).

Im Gegensatz zu FDIR = 1 (Umgeschaltet entsprechend dem vorherigen Zeitschritt) geben die Werte 2, 3 und 4 immer die absolute Flussrichtung im Bauteil an. Weiterhin werden bei FDIR = 2, 3 oder 4 die Ergebnisfelder und -matrizen bzgl. der absoluten Bauteilorientierung angezeigt.
D. h. wird z. B. das kalte Bauteil 119 rückwärts (FDIR = 3) mit einem heißen Medium durchströmt, dann zeigt nach der Simulation die Ergebnis-Temperatur Matrix im rechten Bereich i.d.R. höhere Temperaturen an.

Diese neuen Werte und Anschluss 4 können z. B. im Zsgh. mit Bauteil 170 (Sammelschienenadapter) verwendet werden.

1.2.10. Dampfturbine (Bauteil 122)

- Stodola Korrektur für kleine Stufenzahlen hinzugefügt (FP1OD = 3).
- Matrix MXPCRIT zur Modellierung des kritischen Druckverhältnisses für die Stodolakorrektur hinzugefügt.
- Verfeinerte Austrittsverlustmethode (FSECT = 7) hinzugefügt (ELEP Berechnungen)
- AEN Methode als Alternative zur Stodola Beziehung hinzugefügt (FP1OD = 4). PB = crit_flux(PB,HB)*AEN
- Zusätzliche FIDENT Modi (11,12,13) , welche Feedback am Logic Port 11 geben.
- FGSOD = 3 hinzugefügt. Valve Point wird dabei am Port 11 P eingelesen (nur wenn FIDENT = 0)
- FGSNOZZLETYPE hinzugefügt, um zwischen einfachen Düsen und Lavaldüsen in der Regelradstufe unterscheiden zu können. Bis Version 15 wurden nur einfache Düsen betrachtet.
- Performance Faktoren können nun Werte, Kennlinien, Werte am Logikport 11, Ausdrücke und eine Kombination von allem sein.

- Erweiterung auf Nicht-H_2O Fluide. Dies deaktiviert die SCC Methoden, welche ja auf Regressionsdaten von Wasserdampf-Turbinen basieren. Abhängig von den gewählten Stoffwerten, können auch längere Rechenzeiten auftreten.

1.2.11. Wellendichtung (Bauteil 123)

- Massenstromumkehr kann berechnet werden, wenn Pout > Pin (FDIR = 1)

1.2.12. Wärmetauscher mit Phasenübergang (Bauteil 124)

1.2.12.1. Vorgabe der Wärmeübergangskoeffizienten bei der numerischen Lösung für Mischungen

Das Bauteil 124 benutzt verschiedene Vorgabewerte für die Vorgabe der Wärmeübertragungskoeffizienten. Bei Fluiden vom Typ Wasser-Dampf oder 2-Phasen-Fluid ist es möglich, den Phasenzustand des kompletten Fluids zu bestimmen. Daher werden für diese Fluide, wenn FTYPHX > 0, die Vorgabewerte AL12ECON, AL34ECON, AL12EVAN, AL34EVAN, AL12SUPN, AL34SUPN und die entsprechenden Exponenten verwendet.

Bei den anderen Fluiden (Mischungen) war es bisher unabhängig vom Schalter FTYPHX nur möglich, die Vorgabewerte AL12N, AL34N und die entsprechenden Exponenten zu verwenden. Ab Release 16 sind für solche Fluide die Werte

- AL12PHTN
- EX12PHT
- AL34PHTN
- EX34PHT

vorgesehen. Sollte bei FTYPHX > 0 innerhalb des Wärmetauschers (numerische Lösung FALG = 1) ein Bestandteil der Mischung eine Phasenänderung durchlaufen, werden diese Vorgabewerte statt AL12N, EX12, AL34N, EX34 verwendet. Das kann z. B. bei der Kondensation des Wasseranteils im Gas der Fall sein.

1.2.12.2. Alternative Vorgabe der Wärmeübergangskoeffizienten bei der numerischen Lösung durch Kernel-Expressions

Alternativ zu den Vorgaben der Nominalwerte und Exponenten der Wärmeübertragungskoeffizienten können die Wärmeübertragungskoeffizienten innerhalb des Wärmetauschers (numerische Lösung FALG = 1) mittels Kernel-Expressions

- EALPH12
- EALPH34

berechnet werden. Um zwischen der Vorgabe der Nominalwerte, Exponenten und den Kernel-Expressions zu unterscheiden, sind ab Release 16 die Schalter

- FALPH12
- FALPH34

vorgesehen.

1.2.13. Luftkondensator (Bauteil 127)

- Unterscheidung zwischen drückendem und saugendem Lüfter hinzugefügt (FAIRPATH). Vor Version 16 wurde nur ein drückender Lüfter modelliert.
- Kennlinie CWINDCORR hinzugefügt, um den Einfluss des Windes auf den Luftstrom abzubilden.

1.2.14. Instationärer Trenner (Bauteil 131)

1.2.14.1. Neuer Berechnungsmodus FFU = 2

Das Bauteil 131 konnte bisher nur aus einem Eingangssignal (Anschluss 1) ein Ausgangssignal (Anschluss 2) ermitteln. Mit dem neuen Modus FFU = 2 ist es möglich, auf Basis der gleichen Gleichungen das Eingangssignal aus dem bekannten Ausgangssignal zu berechnen. Das kann z. B. interessant sein, wenn man ein in der Zeit gedämpftes Messsignal zur Verfügung hat und daraus den Zeitverlauf eines nicht gedämpften Wertes ausrechnen möchte. Die Dämpfung kann z. B. durch eine dicke Panzerung eines Messwertsensors verursacht werden.

1.2.15. Schichtenspeicher (Bauteil 145)

1.2.15.1. Zusätzliche Anschlüsse 3, 4 für das Medium

Das Bauteil 145 hat ab Release 16 neue Anschlüsse für das Medium. Bis Release 16 waren Anschlüsse 1 (Medium-Eintritt), 2 (Medium-Austritt) und 3 (Logik-Anschluss für den Betrag der ausgetauschten Wärme während des Zeitschritts) vorhanden. Der Anschluss 1 war beim Entladen im unteren („kalten“) Teil des Speichers und beim Beladen im oberen („warmen“) Teil des Speichers gemeint. Der Anschluss 2 war beim Entladen im oberen („warmen“) Teil des Speichers und beim Beladen im unteren („kalten“) Teil des Speichers gemeint.

Ab Release 16 sind die Anschlüsse wie folgt:

- Anschluss 1 – Medium-Eintritt beim Entladen des Speichers
- Anschluss 2 – Medium-Austritt beim Entladen des Speichers
- Anschluss 3 – Medium-Eintritt beim Beladen des Speichers
- Anschluss 4 – Medium-Austritt beim Beladen des Speichers
- Anschluss 5 – Logik-Anschluss für den Betrag der ausgetauschten Wärme während des Zeitschritts (ehemaliger Anschluss 3)

Der Schalter FCHARGE hat einen neuen Wert 2, um mit den neuen Anschlüssen rechnen zu können. Um mit der alten Logik zu rechnen, kann man entweder mit FCHARGE = 0 oder mit FCHARGE = 1 arbeiten. Bestehende Modelle müssen nicht angepasst werden.

1.2.16. Kolbenmotor (Bauteil 153)

- Erweiterung der Datenstruktur um den maximalen H_2-Gehalt (Vol%) und die Möglichkeit, die Liste der Datensätze danach zu filtern

Zusätzliche Ergebnisvariablen:

- RALAM: Luftverhältnis

1.2.17. Kennfeld basierter Kompressor (Bauteil 159)

- Polytropic-Head Methode für den Wirkungsgrad hinzugefügt.
- Flag zum Berücksichtigen der Kompressibilität bei der Berechnung des korrigierten Massenstroms und der korrigierten Drehzahl (F_USE_Z_REF) hinzugefügt
- Flag zur Behandlung der Wellendrehzahl im IGV Modus (FY = 3) hinzugefügt

1.2.18. Schalter FDPNUM: Handhabung des Druckverlustes in der numerischen Berechnung bei den Bauteilen 7, 10, 61, 119, 124, 126

Bei der numerischen Berechnung (transiente Berechnung in den Bauteilen 7, 10, 61, 119, 126 sowie Berechnung bei FALG = 1 im Bauteil 124) wurde bis Release 15 der Mittelwert des Fluiddruckes zwischen Ein- und Austritt verwendet. Ab Release 16 wird diese Berechnung verfeinert. Dazu wurde in den genannten Bauteile ein neuer Schalter FDPNUM eingeführt.

Bei FDPNUM = 0 wird, wie bisher, mit einem Mittelwert des Fluiddruckes zwischen Ein- und Austritt gerechnet.

Bei FDPNUM = 1 wird dagegen eine lineare Druckverteilung zwischen dem Ein- und Austritt angenommen und mit entsprechenden Druckwerten in den einzelnen Fluidelementen (Anzahl über NFLOW gesteuert) gerechnet.

1.3. Stoffwerte

1.3.1. Effektive Stoffwertfunktionen zur Phasenglättung

Bei der Modellierung des Latentwärmespeichers hat es sich als sinnvoll erwiesen, effektive Stoffwertfunktionen zu entwickeln, die am Phasenübergang einen geglätteten Verlauf aufweisen, da Singularitäten am Phasenübergangspunkt numerische Probleme bereiten und eine Sonderbehandlung solcher Fälle vermieden werden sollte.

Diese Funktionen stimmen außerhalb eines kleinen Bereichs um den Phasenübergangspunkt (0.05 K nach oben und nach unten) mit den realen Stoffwertfunktionen überein. Innerhalb dieses Bereichs erfolgt ein geglätteter Übergang von der einen zur anderen Phase.

Folgende effektive Stoffwertfunktionen stehen zur Verfügung:

- heff (p, T)
- cpeff (p,T)
- rhoeff (p,T)
- lambdaeff (p,T)
- xeff (p,T)
- phaseeff (p,T)

Zusätzlich gibt es die Funktionen

- heff (p, x1, x2): Phasenübergangsenthalpie zwischen dem Zustand x1 und dem Zustand x2
- teff (p,x): liefert die dem Zustand x zugeordnete effektive Temperatur. Beispielsweise erhält man für den Phasenübergang fest (x = 10) / flüssig (x = 11) für x = 10.5 die tatsächliche Phasenübergangstemperatur, für x = 10.0 die untere und für x = 11.0 die obere Grenze des Glättungsbereichs.

1.3.2. PCM-Fluide

PCM-Fluide (engl. **p**hase **c**hange **m**aterial) sind Stoffe, welche die zugeführte oder freigesetzte Energie mit Hilfe eines Phasenwechsels speichern oder abgeben. Für den Phasenwechsel wird Energie benötigt. Dadurch ist ein Großteil der Energie als latente Wärme vorhanden und die Einsatztemperaturbereiche werden kleiner als im Vergleich mit sensiblen Wärmespeichern. Die hier beschriebenen Stoffe durchlaufen einen Phasenübergang von fest zu flüssig (Schmelzen) oder von flüssig zu fest (Erstarren). Der Phasenwechsel findet innerhalb eines Schmelzintervalls statt. Um Berechnungen in dem Zweiphasengebiet durchführen zu können, wird eine effektive spezifische Wärmekapazität definiert, sodass das Integral über das Schmelzintervall der effektiven spezifischen Schmelzenthalpie (Summe aus Schmelzenthalpie und sensibler Enthalpie) entspricht:

$$\Delta h_{eff} = \Delta h_{Sch} + \Delta h_{Sens} = \int_{T_0}^{T_1} (c_{p,eff}(T)dT)$$

Die PCM-Fluide sind in Ebsilon unter den Ölen und Schmelzen zu finden. Insgesamt sind vier vordefinierte Fluide vorhanden. Außerdem besteht die Möglichkeit ein benutzerdefiniertes Fluid auf Basis eines JSON-Formats zu erstellen. Obwohl beim Aufruf der Stoffwertfunktionen der Druck formal angegeben werden muss, sind die Berechnungen ausschließlich von der Temperatur abhängig. Folgende Stoffwertfunktionen stehen zur Verfügung:

- h(p,T)
- h(p,s)
- s(p,T)
- s(p,h)
- t(p,h)
- t(p,s)
- x(p,T)
- x(p,h)
- cp(p,T)
- cp(p,h)
- v(p,T)
- v(p,h)
- eta(p,T)
- eta(p,h)
- lambda(p,T)
- lambda(p,h)
- rho(p,T)
- rho(p,h)
- nue(p,T)
- nue(p,h)
- t_min(p)
- t_max(p)
- t_freeze(p)
- t_melt(p)
- p_min(T)
- p_max(T)
- h_freeze(T)
- h_melt(T)

1.3.2.1 Vordefinierte Fluide

Bei den vordefinierten Fluiden des Herstellers Rubitherm handelt es sich um Stoffe mit einem effektiven Schmelzintervall von ungefähr 15K, wobei die mittlere Temperatur in etwa der Zahl im Markennamen des PCM entspricht. Zusätzlich gibt es untere und obere Schmelzintervalltemperaturen, im JSON-Interface bezeichnet als t_trans_lower und t_trans_upper und die Schmelztemperatur t_melt und die Erstarrungstemperatur t_freeze. Die Schmelzintervalltemperaturen markieren die Grenzen des Temperaturbereichs, in dem die effektive spezifische Wärmekapazität die Anteile der Schmelzenthalpie enthält (siehe oben) und somit ansteigt. Im Schmelzintervall durchläuft das PCM-Fluid einen Phasenübergang von fest nach flüssig. Beim Überschreiten der Schmelztemperatur t_melt wird angenommen, dass das PCM-Fluid im flüssigen Zustand ist (wichtig z. B. bei der Berücksichtigung der Konvektion in Bauteil 166). Beim Unterschreiten der Erstarrungstemperatur t_freeze wird angenommen, dass das PCM-Fluid im festen Zustand ist.

Im Folgenden sind die wichtigsten Eigenschaften der Fluide zusammengefasst:

- RT44HC: Paraffin, Einsatztemperaturen 25-70°C, effektives Schmelzintervall 37-52°C, effektive spezifische Schmelzenthalpie 250,51 kJ/kg, Schmelztemperatur 41°C, Erstarrungstemperatur 44°C
- RT64HC: Paraffin, Einsatztemperaturen 25-95°C, effektives Schmelzintervall 57-72°C, effektive spezifische Schmelzenthalpie 243,66 kJ/kg, Schmelztemperatur 64°C, Erstarrungstemperatur 65°C
- RT80HC: Fettsäure, Einsatztemperaturen 25-110°C, effektives Schmelzintervall 73-88°C, effektive spezifische Schmelzenthalpie 198,11 kJ/kg, Schmelztemperatur 77°C, Erstarrungstemperatur 80°C
- SP58p: Salzhydrat (Natriumacetat-Trihydrat), Einsatztemperaturen 25-80°C, effektives Schmelzintervall 51-66°C, effektive spezifische Schmelzenthalpie 242,42 kJ/kg, Schmelztemperatur 58°C, Erstarrungstemperatur 58°C

1.3.2.2 Benutzerdefinierte Fluide

Hinweis**:** Alle vordefinierten Fluide sind bereits in dem benötigten JSON-Format vorhanden und können durch Copy-and-Paste als Vorlage für benutzerdefinierte Fluide benutzt werden.

Die benutzerdefinierten PCM werden mithilfe des JSON-Formats erstellt. Für die vollständige Funktionalität der PCM werden folgende Angaben vorausgesetzt:

- Name („name“)
- Minimale und maximale Einsatztemperaturen („t_min“ und „t_max“)
- Schmelztemperatur „t_melt“ und Erstarrungstemperatur „t_freeze“ (Solidus- und Liquidustemperatur), wird z. B. bei der Berücksichtigung der Konvektion im Bauteil 166 verwendet
- Die Grenzen des effektiven Schmelzintervalls „t_trans_lower“ und „t_trans_upper“, wird z. B. im Zusammenhang mit „region_polynomial“ und „region_cp_with_h_eff“ benötigt
- Spezifische Wärmekapazität („cp“)
- Dichte („rho“)
- Wärmeleitfähigkeit („lambda“)
- Dynamische Viskosität („eta“)

Außerdem stehen weitere optionale Angaben zur Verfügung:

- Minimale und maximale Einsatzdrücke („p_min“ und „p_max“)
    - Standardmäßig auf 0.001-1000bar gesetzt
- Nullpunktdefinition durch Druck, Temperatur, spezifische Entropie und Enthalpie
    - „t0“, „p0“, „h0“ und „s0“
    - Standardmäßig auf 0 kJ/kg und 0 kJ/(kgK) bei 1bar und 25°C gesetzt

Die meisten Kenngrößen werden mithilfe von primitiven Typen beschrieben. Die physikalischen Stoffeigenschaften Wärmekapazität, Dichte, Wärmeleitfähigkeit und dynamische Viskosität müssen durch verschiedene komplexere JSON-Objekte definiert werden, welche wiederum einen der vordefinierten Typen voraussetzen. Diese Typen werden im Folgenden aufgelistet und die Vorgaben für das JSON-Format beschrieben:

- „step_points“ → sind von den physikalischen Stoffeigenschaften abhängig
    - Wärmekapazität: stückweise konstante Funktion; Punkte definieren Anfang des konstanten Wertes, welche bis zum nächsten Punkt gilt
        - Vorteil: Enthalpie wird stückweise lineare Funktion
    - Andere Stoffeigenschaften: stückweise lineare Funktion zwischen dem Anfangs- und Endpunkt
    - Definiert als ein Array aus Punkten, welche wiederum Arrays sind
- „polynomial“
    - Gesamter Einsatztemperaturbereich wird durch ein einzelnes Polynom beliebigen Grades definiert
    - Definiert als ein Array aus den Koeffizienten des Polynoms, angefangen mit dem kleinsten Koeffizienten a_0 (Absolutglied)
- „region_polynomial“
    - Die feste und die flüssige Phase werden jeweils mit einem eigenen Polynom beschrieben; das Schmelzintervall wird linear interpoliert
    - Definiert als ein Objekt mit den Eigenschaften „liquid“ und „solid“, welche den jeweiligen Arrays der Polynomkoeffizienten zugewiesen werden
- „piecewise_polynomial“
    - Gesamter Einsatztemperaturbereich wird durch beliebig viele Polynome definiert
    - Gesamter Einsatztemperaturbereich muss abgedeckt und die Polynome nach dem Definitionsbereich geordnet angegeben werden
    - Besteht aus einem Array der stückweisen definierten Polynome
    - Polynome definiert als Objekt mit oberer und unterer Grenze des Definitionsbereichs („t_min“ und „t_max“) sowie dem Array der Polynomkoeffizienten („polynomial“)
- „expression“
    - Definition einer beliebigen temperaturabhängigen Funktion mithilfe aller gängigen Funktionen und Operatoren durch einen einfachen String
    - Verwendung des Fragezeichenoperators für Definition unterschiedlicher Funktionen in Abhängigkeit des Temperaturbereichs
        - condition ? true case : false case
- „region_cp_with_h_eff“ → nur für die Wärmekapazität verfügbar
    - Feste und flüssige Phase durch einen der vorherigen Typen definiert („region_polynomial“ ausgenommen)
        - „solid“ und „liquid“ als Objekte mit jeweiligen Typ
    - Verlauf im Zweiphasengebiet wird durch Angabe der effektiven spezifischen Schmelzenthalpie („h_eff“) berechnet
        - Funktion im Zweiphasengebiet: $c_p\,(T) = a + b \cdot (1\text{-}tanh^2(c \cdot T + d))$
        - c kann vom Benutzer vorgegeben werden (standardmäßig 0.26)
        - Die restlichen Koeffizienten werden aus den Randbedingungen ermittelt
            - → Δh_{eff} eingehalten und
            - → Stetigkeit der Funktion beiden Rändern

Es folgt eine syntaktisch korrekte Definition eines PCM-Fluids, bei dem jeder der vorhandenen Typen benutzt wird. **Es handelt sich hierbei nur um ein Beispiel der verschiedenen Typen, und nicht um ein reales Fluid für Berechnungen.**

```
{
	„name“ : „MyFluid“,
	„p_min“ : 0.1,
	„t_min“ : 20,
	„t_max“ : 60,
	„t_melt“ : 38,
	„t_freeze“ : 38,
	„t_trans_lower“ : 35,
	„t_trans_upper“ : 45,
	„t0“ : 35,
	„p0“ : 1,
	„h0“ : 1000,
	„s0“ : 0.8,
	„cp“ : {
		„region_cp_with_h_eff“ : {
			„h_eff“ : 250,
			„c“ : 0.4,
			„liquid“ : {
				„step_points“ : [
					[45, 1.6],
					[50, 1.7],
					[60, 1.8]
				]
			},
			„solid“ : {
				„expression“ : „t>30 ? 1.5E-2*t : sin(t+3)“
			}
		}
	},
	„rho“ : {
		„polynomial“ : [1700, 3.5, 4E-06]
	},
	„lambda“ : {
		„region_polynomial“ : {
			„liquid“ : [0.2],
			„solid“ : [0.1, 0.004]
		}
	},
	„eta“ : {
		„piecewise_polynomial“ : [
			{
				„t_min“ : 45,
				„t_max“ : 55,
				„polynomial“ : [30]
			},
			{
				„t_min“ : 55,
```

```
                                    „t_max“ : 60,
                                    „polynomial“ : [40]
                                }
                        ]
                }
}
```

1.3.3. Helisol-Fluide

Die Thermoöle Helisol 5A, Helisol XA und Helisol XLP der Firma Wacker wurden in die Standard-Datenbank aufgenommen, jeweils in einer ungebrauchten und einer gebrauchten (d. h. nach 750 Betriebsstunden) Variante und jeweils für die vier Druckstufen 1 bar, 10 bar, 20 bar und 30 bar.

Im Gegensatz zu den bisher integrierten Thermoölen sind die Helisol-Daten nicht mit Polynomen, sondern mit Kennlinien interpoliert.

Bei der Berechnung wird nur die Abhängigkeit von der Temperatur, nicht aber die Abhängigkeit vom Druck betrachtet. Diese kann nur durch die Auswahl des am besten geeigneten Datensatzes berücksichtigt werden. Eine Plausibilitätsprüfung mit Hinblick auf den Leitungsdruck findet nicht statt.

1.3.4. TREND wird zu TREND 4. Neu TREND 5

Der Leitungstyp „Universalfluid“ ermöglicht die Verwendung der Stoffwertbibliothek „TREND“ der Ruhr-Universität Bochum. Hier unterstützt Ebsilon 15 die Version 4. In Ebsilon 16 wird zusätzlich die Version 5 unterstützt. Um diese unterscheiden zu können, wurde die Bibliothek „TREND“ in „TREND 4“ umbenannt. Neu hinzugekommen ist die Bibliothek „TREND 5“.

TREND 5 unterstützt folgende zusätzliche Stoffe:

- 1-Hexene (Hexylene)
- POE5 (pentaerythritol tetrapentanoate)
- POE7 (pentaerythritol tetraheptanoate)
- POE9 (pentaerythritol tetranonanoate)

Bei Modellen, die mit Versionen vor Release 16 gespeichert sind, werden „TREND“ Biblotheken als „TREND 4“ geladen. Beim Wechsel von TREND 4 nach 5 und zurück bleiben möglichst alle Zusammensetzungen und Optionen erhalten. (siehe auch Kapitel 1.3.5).

1.3.5. Bibliothekswechsel bei Universalfluid

Wenn beim Leitungstyp „Universalfluid“ eine Bibliothek gewechselt wird, dann wird nun die Zusammensetzung beibehalten (natürlich nur für Stoffanteile, die in der neuen Bibliothek verfügbar sind).

1.3.6. Neue Optionen bei der Zusammensetzungsdefinition

Bei den Leitungstypen Luft, Rauchgas, Öl, Kohle, Rohgas und Benutzerdefinierte-cp-Koeffizienten (auch als FDBR-Leitungstypen bekannt) kann nun bei jedem Leitungstyp jede der Zusammensetzungsdefinitionen verwendet werden.

Zusätzlich gibt es vier neue Zusammensetzungsdefinitionen:

- Masse ohne Feststoffe
- Mol ohne Feststoffe
- Trockene Masse ohne Feststoffe
- Trockene Mol ohne Feststoffe

wobei Feststoffe die folgenden Stoffe sind: C, H, O, N, S, CL, ASH, LIME, CA, H2OB, ASHG, MG, CACO3, CAO, CASO4, MGCO3 und MGO

Die Feststoffe werden als partielle Massenbilanz vorgegeben / angezeigt, und die restlichen Stoffe werden (ggfs. ohne H_2O) als Gemisch auf 100 % skaliert (als Massen- bzw. Molzusammensetzung).

Diese neuen Optionen können selbstverständlich auch bei Darstellung von Zusammensetzungen in Wertekreuzen und bei Aufrufen in EbsScript und EbsOpen verwendet werden.

2. Benutzeroberfläche

2.1. Neue Datentypen für EbsMatrix

Bisher konnten in Objekten vom Typ EbsMatrix (Vorgabe- bzw. Ergebnismatrizen) ausschließlich Fließkomma-Zahlenwerte (Datentyp „double“) gespeichert werden.

Benutzerdefinierte Matrizen in Makroobjekten können nun als Größe für den Wert zusätzlich einen der folgenden Datentypen haben:

- „Integer“ (ganzzahlige Werte)
- „Text“ (Text)
- „Variant“ (jede Zelle kann unabhängig von den anderen einen Fließkomma-Zahlenwert, einen ganzzahligen Wert oder einen Text enthalten)

2.2. Innerer Rand bei Textfeldern

Bei Textfeldern kann nun für alle vier Richtungen (oben, unten, rechts und links) ein Abstand in logischen Punkten zur äußeren Begrenzung angegeben werden. Dies ermöglicht z. B. bei einem gerahmten Textfeld den Abstand des Textes zum Rahmen zu kontrollieren.

2.3. Simulationsbenachrichtigungen manipulieren

In den Modell-Einstellungen unter „Simulation“ -> „Benachrichtigungseinstellungen“ können Fehler, Warnungen und Kommentare manipuliert werden. D. h. sie können einer anderen Stufe (Fehler / Warnung / Kommentar) zugewiesen oder auch vollständig ausgeblendet werden.

Hierzu kann in der Liste in den linken vier Spalten jeweils eine Auswahl getroffen werden (Fehlernummer, Objekttyp, minimaler und maximaler Fehlerlevel), und in der letzten Spalte ein neuer Fehlerlevel zugewiesen werden.

ACHTUNG: für jede Meldung wird die Liste von oben nach unten durchsucht und die **erste Zeile**, die die Meldung erfüllt legt den neuen Fehlerlevel fest.

Wenn als neuer Fehlerlevel „Callback“ ausgewählt ist, dann wird der unten angegebene EbsScript-Ausdruck mit der aktuellen Meldung aufgerufen und dessen Ergebnis als neuer Fehlerlevel verwendet. Die Callback-Ausdruck muss vom Typ

function (const error: CALCULATIONERROR) : NOTIFICATIONSETTING;

sein.

Die Typen CALCULATIONERROR und NOTIFICATIONSETTING sind beide in der Standard-Unit @System definiert.

Zum Beispiel ändert folgender Funktion EbsScript-Ausdruck

```
function (const error: CALCULATIONERROR) : NOTIFICATIONSETTING
begin
	if error.errorLevelOriginal = errorLevelWarning then
	begin
		result.errorLevel := errorLevelComment;
		result.apply := true;
	end;
end
```

alle Warnungen in Kommentare. (Hinweis: Es handelt sich hierbei um eine ***unbenannte Funktion*** oder auch ***Lambda-Funktion***. Diese hat keinen Namen und auch kein Semikolon zwischen Rückgabetyp und dem darauf folgenden begin.)

Alternativ können die Benachrichtigungseinstellungen auch komponentenspezifisch in den Komponenten-Eigenschaften auf der Seite „Benachrichtigungseinstellungen" bearbeitet werden. Beim Durchsuchen werden immer zuerst die komponentenspezifischen Benachrichtigungseinstellungen vor den auf Modellebene spezifizierten behandelt.

Hinweis: In der Fehlerleiste unter „Extras" können mit der Option „Benutzermodifikationen abschalten (originale Meldungen anzeigen)" etwaige Modifikationen abgeschaltet und die originalen Meldungen dargestellt werden.

2.4. Simulations-Dialog

In den allg. Einstellungen unter „Berechnung" -> „Zeige Messagebox während / nach Berechnung" gibt es eine neue Option: „Simulationsfortschritt anzeigen" (diese ist nun auch die Voreinstellung bei neuen Modellen).

Bei dieser Option erscheint während der Simulation ein Dialog, der den Konvergenzverlauf anzeigt und auch das vorzeitige Abbrechung einer Simulation ermöglicht. Hierbei kann auch die maximale Iterationszahl nachträglich nach oben und unten angepasst werden.

2.5. Benutzerdefinierte Toolbar / Menü-Befehle

In Release 16 können in der Oberfläche Menüpunkte und Toolbarbuttons mit benutzerdefinierten Befehlen ausgestattet werden.

2.5.1. Benutzerdefinierte Befehle in Toolbar / Menü einfügen

Zum Einfügen eines benutzerdefinierten Befehls wählen Sie „Anpassen" der Menüs / Toolbars (Extras -> Anpassen...). Im „Anpassen"-Dialog unter „Befehle / Kategorien" wählen Sie „Neues Kommando". Im „Befehle"-Fenster erscheint dann auch „Neues Kommando", welche Sie auf einen Toolbar oder in ein Menü mittles Drag & Drop ziehen können.

Aktivieren Sie nun auf dem neu eingefügten Button / Menüpunkt das Kontextmenü und wählen Sie „Schaltflächendarstellung...“. Neben Bezeichung und Bild gibt es nun auch ein „Knopf-Kommando“. Hier wird der auszuführende Befehl als JSON kodiert angegben. Folgende Syntax / Schlüsselworte werden hier verwendet:

```
{
„command“:
        {
                „id“:„Befehls-Id“,
                „data“:Daten
        }
}
```

Mögliche Befehle:

Befehls-Id	**Daten**	**Auswirkung**
none		Keine Auswirkung
simulate		Simulation ausführen
validate		Validierung ausführen
ebsscript	{„argument“:*ID*, „param_int“:*Integer-Wert*, „param_string“:„*String-Wert*“}	EbsScript mit geg. ID ausführen
shiftview	{„argument“:*ID*}	Wartenansicht aufschalten
provismt	{„argument“:*ID*}	Provis Multitrend
provisxy	{„argument“:*ID*}	Provis xy-Trend
provismultitimetrend	{„argument“:*ID*}	Provis Multizeittrend
proffirstchild		Erstes Kind Profil aktivieren
proflastchild		Letztes Kind Profil aktivieren
profparent		Eltern Profil aktivieren
profnextsibl		Nächstes Geschwisterprofil aktivieren
profprevsibl		Vorheriges Geschwisterprofil aktivieren
profnext		Nächstes Profil aktivieren
profprev		Vorheriges Profil aktivieren
profid	{„argument“:*ID*}	Profil mit geg. ID aktivieren
profname	{„argument“:„*Name*“}	Profil mit geg. Namen aktivieren
provistrend	{„argument“:*ID*}	Provis Trend
ebsscriptextern	{„argument“:„*Pfad*“, „param_int“:*Integer-Wert*, „param_string“:„*String-Wert*“}	Externes EbsScript ausführen
insert_partial_model	{„tag“:„*Tag-Name*“, „name“:„*Makro-Name*“}	Teilschaltung / Macro einfügen

ebsscript_direct	{„code“:„*EbsScript-Code*“}	Angegebenen EbsScript-Code ausfüren
open_model	{„path“:„*Name/Pfad*“}	Modell mit geg. Namen / Pfad öffnen
open_context	{„context“:„*Name*“}	Aktiviert im aktiven Model den angegeben Kontext

multi_command	{„data“:[
	{*Befehl*},...
	]}
	
	Wobei jeder {*Befehl*} folgendermaßen strukturiert sein muss:
	
	{
	„command“:
	{
	„id“:„*BefehlsID*“,
	„data“:*Daten*
	},
	„on_error_continue“:boolean,
	„on_success_continue“:boolean,
	„negate_command_result“:boolean
	}
		Führt mehrere Befele aus.
		Die Felder
		„on_error_continue“,
		„on_success_continue“ und
		„negate_command_result“
		kontrollieren die Ausführung mehrerer Befehle und sind alle optional.
		Das Feld „on_success_continue“ hat als Defaultwert true.
		Die Felder „on_error_continue“ und „negate_command_result“ haben als Defaultwert false.

Beispiel: Befehl zum Einfügen des Ebsilon-Makros „Diagnosis_Coloring“

```
{
„command“:
        {
                „id“:„insert_partial_model“,
                „data“:{„tag“:„Ebsilon“:„name“:„Diagnosis_Coloring“}
        }
}
```

2.5.2. Befehle zum Einfügen von Teilschaltungen (Makros)

Beim „Anpassen“ der Menüs / Toolbars (Extras -> Anpassen...) können im „Anpassen“-Dialog unter „Befehle / Kategorien“ Befehle zum Einfügen von Makros ausgewählt werden.

2.6. Diagramme

2.6.1. Zeichnen der Hintergrund-Iso-Kurven: automatische Anpassung des Suchbereichs an die jeweilige Stoffwerttafel

Beim Zeichnen der Hintergrund-Iso-Kurven in den Diagrammen vom Typ „h,s“, „t,s“, „log p,h“ wurden bisher X- und Y-Werte in einem statisch definierten Bereich gesucht. Dieser Bereich passte gut für die Wasser-Dampf-Tafel, jedoch weniger gut für z. B. Wasserstoff. Daher wurde die Suche ab

Release 16 verfeinert. Zumindest bei den Fluiden, wo es möglich ist, wird versucht zur Laufzeit die Minimum- und Maximum-Werte für die Suche direkt in der jeweiligen thermodynamischen Funktion zu ermitteln.

Diese Grenzwerte für die Suche sind **nicht zu verwechseln** mit den benutzerdefinierten Achsenskalierungen / Zoom-Einstellungen.

2.6.2. Möglichkeit der Auswahl der Achsen-Einheit

Ab Release 16 ist es möglich, eine nicht Standard-EBSILON-Einheit für die X-Achse oder die Y-Achse des Diagramms auszuwählen. Dazu gibt es im Diagramm-Dialog unter Menü Einstellungen – Eigenschaften für die jeweilige Achse (X und Y) eine Combobox „Achseneinheit".

2.6.3. Fadenkreuz-Cursor

In Diagrammen kann nun auch ein Fadenkreuz-Cursor verwendet werden, der die aktuelle Position und Informationen über Linien anzeigt. Der Fadenkreuz-Cursor kann über das Kontextmenü an- und ausgeschaltet werden.

2.7. Leitungswerte in selektiertes Bauteil übernehmen

Für Bauteil 1 (Randwert) und 33 (Startwert) besteht die Möglichkeit, sie an (1) bzw. auf (33) eine bereits mit Daten gefüllte Leitung zu platzieren und dann im Kontextmenü (rechte Maustaste) die Funktion „Leitungswerte in selektiertes Bauteil übernehmen" auszuführen.

Der Schalter FNCVSRC, der angibt, ob der Heizwert aus der Zusammensetzung berechnet oder manuell vorgegeben werden soll, kann allerdings nicht aus der Leitung übernommen werden, da in der Leitung nur der Heizwert selbst gespeichert ist, nicht aber die Information über die Quelle des Heizwertes.

Aus diesem Grunde wurde bis Release 15 der Schalter FNCVSRC unverändert gelassen. Allerdings stellte dies eine potentielle Fehlerquelle dar, da die Bauteile 1 und 33 häufig durch Kopieren von einer anderen Stelle der Schaltung eingefügt werden und FNCVSRC dann den Wert beibehielt, den es an der anderen Stelle hatte, und dadurch möglicherweise mit einem falschen Heizwert gerechnet werden konnte, ohne dass es bemerkt wurde.

In Release 16 erscheint deshalb eine Messagebox, in der der Anwender festlegen muss, welche Einstellung der Schalter FNCVSRC bekommen soll.

Auch der Schalter FFLSET ist nicht auf der Leitung vorhanden. Allerdings ist hierbei das Gefahrenpotential nicht so hoch, da eine falsche Setzung von FFLSET zu einer Fehlermeldung führt.

3. Zusatzmodule

3.1. EbsScript

3.1.1. Parsen von Digraphen

Bei Digraphen handelt es sich jeweils um zwei Zeichen, die als ein zusammenhängendes Token verarbeitet werden. Beispiele hierfür sind „:=“ (Zuweisung) und „<=“ (kleiner gleich). In früheren Ebsilon-Versionen wurden zwei zueinander passende Zeichen auch dann als Digraph geparst, wenn sie durch sog. Whitespace (Leerzeichen, Tabulatoren, Zeilenumbrüche) oder sogar Kommentare getrennt waren.

In Standard Pascal und auch Delphi ist dies nicht erlaubt, daher ist das Ziel, in einer künftigen Ebsilon Version Digraphen nur noch dann zu erkennen, wenn diese direkt hintereinanderstehen.

Um die Umstellung zu erleichtern, dürfen ab Release 16 Digraphen höchstens noch durch Whitespace voneinander getrennt sein (keine Kommentare mehr dazwischen erlaubt). Zudem MUSS der Digraph „::“ zusammenhängend geschrieben werden. Dies ist notwendig, um die Zeichenfolgen „::␣:“ und „:␣::“ unterscheiden zu können.

Des Weiteren gibt es im EbsScript-Editor unter „Extras->Optionen...->Syntax“ über die Einstellung „Relaxiertes Parsen von Digraphen“ die Möglichkeit, das Verhalten des Parsers zu ändern: Ist „Relaxiertes Parsen von Digraphen“ angeschaltet, dann darf Whitespace zwischen den Zeichen vorkommen, andernfalls nicht. Das Verhalten kann auch über das Pragma ${RDP(+)} bzw. ${RDP(-)} im Quellcode geändert werden.

3.1.2. Vordefinierte Kompilierzeit-Macros

Mit dieser Release werden ***Kompilierzeit-Macros*** eingeführt. Diese werden **während des Kompilierens** durch die jeweiligen **Konstanten** ersetzt. D. h. alle Werte werden zum Zeitpunkt der Kompilation bestimmt und ändern sich danach nicht mehr. Der Zugriffssyntax lautet:

$(*Macroname*)

Folgende vordefinierten Macros stehen zur Verfügung:

Macro	**Typ**	**Beschreibung**
$(file)	string	Name der kompilierten Datei
$(line)	integer	Aktuelle Zeilennummer
$(date)	string	Aktuelles Datum (Zur Kompilation!)
$(time)	string	Aktuelle Zeit (Zur Kompilation!)
$(program)	boolean	Ist kompilierter Code ein Hauptprogram?
$(unit)	boolean	Ist kompilierter Code eine Unit?
$(subroutinename)	string	Name der kompilierten Funktion / Prozedur
$(subroutinefullname)	string	Name der kompilierten Funktion / Prozedur mit Typbeschreibung („function“/„procedure“, Name, Argumentliste und Rückgabetyp)

$(version)	string	Ebsilon-Version (z. B. 16.1.0.30567)
$(versionmajor)	integer	Ebsilon-Release-Nummer (z. B. 16)
$(versionpatch)	integer	Ebsilon-Patch-Nummer (z. B. 1)
$(versionhotfix)	integer	Ebsilon-Hotfix-Nummer (z. B. 0)
$(versionrevision)	integer	Ebsilon-Revisions-Nummer (z. B. 30567)
$(counter)	integer	Zähler: startet bei 0 und erhöht sich bei jedem Aufruf um 1. Startet für jede einzelne Übersetzungseinheit (Programm, Unit) bei 0

3.1.3. Neuer primitiver Typ „Byte“

Es gibt einen neuen primitiven Typen namens „Byte“. Variable und Konstanten von diesem Typ entsprechen einem 8 Bit breiten vorzeichenlosen Integer, d. h. sie nehmen Werte von 0 - 255 an. Alle arithmetischen Rechnungen finden modulo 256 statt.

Bitte beachten Sie, dass im Gegensatz dazu der Typ „Char“ einem 16 Bit breiten Integer entspricht und einem UCS-2 (UTF-16) Zeichen entspricht (EbsScript-Strings sind UCS-2 kodiert.)

3.1.4. Inline Definition von Record-Instanzen

Instanzen von Record-Typen können nun als Konstanten inline im Code definiert werden. Die Syntax hierfür ist:

Typname(feld1:wert1; feld2:wert2; ...)

z. B.

```
type person_type=record
        last_name:string;
        middle_name:string;
        first_name:string;
end;

var person:person_type;
begin
        //...
        person:=person_type(last_name:„Miller“; first_name:„Arthur“);
        //...
end;
```

Wie bei der Definition von Record-Konstanten müssen die Felder in der Initialisierung in der gleichen Reihenfolge wie in der Definition des Records vorkommen, es dürfen aber Felder ausgelassen werden (diese werden Default-Initialisiert).

ACHTUNG: Diese Erweiterung gilt nur für Record-Typen, NICHT für Class-Typen.

3.1.5. „case ... of“ Erweiterungen

Der Typ bei „case .. of“ kann nun auch ein String sein, dabei müssen die einzelnen Ansprungstellen Stringkonstanten sein:

```
procedure foo(s:string);
begin
        case s of
        „Hallo“ :
        begin
                // ...
        end;

        „Welt“ :
        begin
                // ...
        end

        otherwise
        begin
                // ...
        end
        end;
end;
```

Der Vergleich findet IMMER Case-Sensitiv statt (Groß / Kleinschreibung wird unterschieden).

Weiterhin können nun als Ansprungstellen ganze Bereiche angegeben werden.

- 5 .. 10 : // alle Werte größer gleich 5 und kleiner gleich 10
- .. 2 : // alle Werte kleiner gleich 2
- 15 .. : // alle Werte größer gleich 15

```
procedure bar(value:integer);
begin
        case value of
        ..-10 :
        begin
                println(„<= -10“);
        end;
        20.. :
        begin
                println(„=> 20“);
        end;
        5..8 :
        begin
                println(„5 <= .. <= 8“);
        end;
```

```
        4 :
        begin
                println(„== 4“);
        end;
        0 :
        begin
                println(„== 0“);
        end
        otherwise
        begin
                println(„something else“);
        end
        end;
end;
```

Bei Strings wird hierbei die lexikographische Ordnung verwendet.

3.1.6. Multiples Implementieren von Interfaces

Eine EbsScript-Klasse (NUR „Class“, KEINE „Record“!) kann nun mehrere Interface gleichzeitig implementieren. Hier z. B. die Definition eines Amphibienfahrzeugs, welches schwimmen und rollen kann:

```
type ICanSwim = interface
        procedure swim;
end;

type IHasWheels = interface
        procedure roll;
end;

type AmphibiousVehicle = class(ICanSwim, IHasWheels)
        procedure swim;override;
        procedure roll;override;
end;
```

Sollte es beim multiplen Implementieren von Interfaces zu Namenskonflikten kommen, dann kann mittels der Methodenzuordnungsklausel die Standardzuordnung außer Kraft gesetzt werde:

```
type ICanSwim = interface
        procedure swim;
        procedure honk;
end;

type IHasWheels = interface
        procedure roll;
        procedure honk;
end;
```

```
type AmphibiousVehicle = class(ICanSwim, IHasWheels)
        procedure swim;override;
        procedure roll;override;

        procedure blow_foghorn; virtual;        // honking when swimming
        procedure trot;     virtual; // honking on wheels

        procedure ICanSwim.honk = blow_foghorn;
        procedure IHasWheels.honk = trot;

end;
```

3.1.7. Schlüsselworte „is“ and „as“

Die Schlüsselworte „is“ und „as“ werden im Zusammenhang mit EbsScript-Klassen (NUR „Class“, KEINE „Record“!) und Interfaces verwendet.

Das Schlüsselwort „is“ ermöglicht zu testen, ob der Laufzeittyp einer Klassen- oder Interfaceinstanz vom Typ einer bestimmte Klasse ist oder ein bestimmtes Interface implementiert. (Sämtliche Kombinationen von Klassen und Interfaces sind hierbei zulässig.)

Folgendes Programm zeigt mögliche Verwendungen von „is“ und „as“:

```
type
        IAnimal = interface
                function make_noise          : string;
        end;

type Dog = Class (IAnimal)
                function make_noise          : string; override;
                procedure tell(command:string);
        end;

function Dog.make_noise      : string;
begin
        result:=„woof“;
end;

procedure Dog.tell(command:string);
begin
        println(„Executing command: „, command);
end;

type Cat = Class (IAnimal)
                function make_noise          : string; override;
                procedure relax();
        end;
```

```
function Cat.make_noise        : string;
begin
        result:=“purr“;
end;

procedure Cat.relax;
begin
        println(„relaxing...“);
end;

procedure interact(animal:IAnimal);
begin
        println(animal.make_noise());

        if animal is Dog then
        begin
                (animal as Dog).tell(„Sitz!“);
        end;

        if animal is Cat then
        begin
                (animal as Cat).relax();
        end;

end;

var
        a_cat:Cat;
        a_dog:Dog;
begin
        a_cat:= Cat.Create();
        interact(a_cat);
        a_dog:= Dog.Create();
        interact(a_dog);
end;
```

Ausgabe des Programms:

```
purr
relaxing...
woof
Executing command: Sitz!
```

3.1.8. Überladen von Klassen-Operatoren

Das Überladen von impliziten und expliziten Operatoren wurde vollständig überarbeitet und bisher fehlende Operatoren ergänzt. Nun sind folgende Operatoren für Klassen überladbar:

Operator	Kategorie	Deklarationssignatur	Symbolzuordnung
Implicit	Konvertierung	Implicit(a : Typ): Ergebnistyp;	Implizite Typumwandlung
Explicit	Konvertierung	Explicit(a: Typ): Ergebnistyp;	Explizite Typumwandlung
Negative	Unär	Negative(a: Typ): Ergebnistyp;	-
Positive	Unär	Positive(a: Typ): Ergebnistyp;	+
Inc	Unär	Inc(a: Typ): Ergebnistyp;	Inc
Dec	Unär	Dec(a: Typ): Ergebnistyp	Dec
LogicalNot	Unär	LogicalNot(a: Typ): Ergebnistyp;	not
Trunc	Unär	Trunc(a: Typ): Ergebnistyp;	Trunc
Round	Unär	Round(a: Typ): Ergebnistyp;	Round
Equal	Vergleich	Equal(a: Typ; b: Typ): Boolean;	=
NotEqual	Vergleich	NotEqual(a: Typ; b: Typ): Boolean;	<>
GreaterThan	Vergleich	GreaterThan(a: Typ; b: type) Boolean;	>
GreaterThan-OrEqual	Vergleich	GreaterThanOrEqual(a: Typ; b: Typ): Boolean;	>=
LessThan	Vergleich	LessThan(a: Typ; b: Typ): Boolean;	<
LessThan-OrEqual	Vergleich	LessThanOrEqual(a: Typ; b: Typ): Boolean;	<=
Add	Binär	Add(a: Typ; b: Typ): Ergebnistyp;	+
Subtract	Binär	Subtract(a: Typ; b: Typ): Ergebnistyp;	-
Multiply	Binär	Multiply(a: Typ; b: Typ): Ergebnistyp;	*
Divide	Binär	Divide(a: Typ; b: Typ): Ergebnistyp;	/
IntDivide	Binär	IntDivide(a: Typ; b: Typ): Ergebnistyp;	div
Modulus	Binär	Modulus(a: Typ; b: Typ): Ergebnistyp;	mod
LeftShift	Binär	LeftShift(a: Typ; b: Typ): Ergebnistyp;	shl
RightShift	Binär	RightShift(a: Typ; b: Typ): Ergebnistyp;	shr
LogicalAnd	Binär	LogicalAnd(a: Typ; b: Typ): Ergebnistyp;	and
LogicalOr	Binär	LogicalOr(a: Typ; b: Typ): Ergebnistyp;	or
LogicalXor	Binär	LogicalXor(a: Typ; b: Typ): Ergebnistyp;	xor
BitwiseAnd	Binär	BitwiseAnd(a: Typ; b: Typ): Ergebnistyp;	bitwiseand
BitwiseOr	Binär	BitwiseOr(a: Typ; b: Typ): Ergebnistyp;	bitwiseor
BitwiseXor	Binär	BitwiseXor(a: Typ; b: Typ): Ergebnistyp;	bitwisexor

3.1.9. Neue und erweiterte Funktionen

- Funktion „getFluxDirectionAtLink“ hinzugefügt: gibt die Flussrichtung zu einem Bauteilanschluss zurück.

- Funktion „getLineTypeAtLink“ für auf Leitungen dockende Bauteile (Typen 33, 45, 46, 105 und 147) verwendbar gemacht: falls das Bauteil auf einer Leitung andockt und für „linkNo“ der Wert 1 angegeben wird, dann wird der Typ der Leitung zurückgegeben.

- Funktionen
  „getProductVersionReleaseNumber“
  „getProductVersionPatchNumber“
  „getProductVersionHotfixNumber“
  „getProductVersionRevisionNumber“
  Diese Funktionen geben die einzelnen Positionen der Versionsnummer (z. B. 15.4.0.29840) als Integer Werte zurück.

- Funktion „runEbsScript“ für Macro-Objekte hinzugefügt. Zum expliziten Ausführen der Macro-EbsScripte „Vor der Berechnung“, „Nach der Berechnung“ und „Übernahme der Nominalwerte“.

- Funktion „getCalculationErrors“ hinzugefügt: gibt die Fehler / Warnungen / Kommentare der letzten Rechnung auf dem aktuellen Profil zurück.

- Methode „getobjects(ebsobjecttype:string, includeChildContexts:boolean)“ zum Typ „ebsmacrobase“ hinzugefügt: erlaubt Zugriff auf Objekte innerhalb eines Macros (ggfs. gefiltert nach Typ bzw. ob auch Objekte aus enthaltenen Macros enthalten sein sollen).

- Eigenschaft „shape:integer“ zum Typ „ebscomp“ hinzugefügt: erlaubt Lesen und Setzen der aktuellen Darstellungsform.

- Eigenschaft „shapeCount:integer“ (nur Lesen) zum Typ „ebscomp“ hinzugefügt: erlaubt Lesen der Anzahl unterschiedlicher Darstellungsformen.

- Prozedur „ksSetComponentCalcState (component:ebscomp; enable_calculation:Boolean; recursive:Boolean = False)“ in der Standard Unit @KernelScripting hinzugefügt: Diese Funktion erlaubt es, während der Simulation die Berechnung eines Bauteils temporär abzuschalten. Temporär abgeschaltete Bauteile behalten dabei die Gleichungen aus dem vorherigen Iterationsschritt. Wenn „component“ ein Macro-Objekt ist, können mit „recursive=True“ alle Objekte innerhalb des Macros temporär ein- bzw. ausgeschaltet werden.
  Bitte beachten Sie, dass es sich hierbei um eine sehr fortgeschrittene Funktionalität handelt, die bei fehlerhafter Verwendung zu schwer nachvollziehbaren Berechnungsfehlern führen kann.

- Mathematische Funktionen
  „trunc(arg:Real):Real“ : gibt den ganzzahligen Anteil einer reellen Zahl zurück, mit dem gleichen Vorzeichen, wie das von „arg“
  „frac(arg:Real):Real“ : gibt den Nachkommateil einer reellen Zahl zurück, mit dem gleichen Vorzeichen, wie das von „arg“

3.1.10. Neue EbsScript Standard-Unit @KernelInterface

Die neue EbsScript Standard-Unit @KernelInterface fasst Funktionen aus den Standard-Units @KernelScripting und @KernelExpression zusammen, welche unabhängig von der aktuellen Berechnungsdomäne (z. B. KernelScript oder KernelExpression) verfügbar sind. Diese Funktionen können von beliebigen EbsScripten während einer Simulation / Validierung aufgerufen werden. Mittels der Funktion „kiGetMode“ kann festgestellt werden, ob und in welchem Iterationsmodus sich die Applikation gerade befindet.

3.1.11. Schlüsselwort- und Objekt-Baum

Am Kopf beider Baumdarstellungen befindet sich nun ein Textfeld, mit dem die Anzeige des Baum gefiltert werden kann. Weiterhin gibt es rechts davon einen Dropdown-Button, der Filteroptionen anzeigt.

So lange keine Großbuchstaben eingegeben werden, findet die Suche unabhängig von der Groß- / Kleinschreibung statt. Sobald mindestens ein Großbuchstabe eingegeben ist, wird bei der Suche Groß- / Kleinschreibung beachtet. Beginnt der Suchtext mit Anführungszeichen (“), dann muss der Text mit den Suchtext beginnen, endet der Suchtext mit Anführungszeichen, dann muss der Text mit den Suchtext enden.

Bei den Funktionen werden nun sämtliche Funktionen aus allen internen Units und direkt in die Sprache eingebaute angezeigt.

Im Objektbaum wird zudem der Punkt (.) verwendet, um in den einzelnen Ebenen des Baumes zu suchen. Mit jedem Punkt wird ein Suchtext für die nächste Ebene erstellt. Für jede Ebene kann so ein Suchtext erstellt werden (der z. B. auch Anführungszeichen enthalten kann). Die Option „Namensräume ignorieren“ ignoriert Macro-Namensräume bei der Suche, d. h. alle Objekte (auch die in Macros) werden in der obersten Ebene gesucht.

3.2. EbsOpen

In der Header-Datei „EbsUser.h“ gibt es eine neue Version des Rechen-Interfaces.

Die Version 10 (im Namespace „v10“, nur für C++ verfügbar) beinhaltet eine CallBack-Funktion zur Berechnung von Stoffeigenschaften. Dies macht das Linken zu wdt.lib und mixture.lib obsolet.

3.2.1. Werte von Ergebniswerte / -felder / -scharen und -matrizen

Die Werte von Ergebniswerten / -feldern / -scharen und -matrizen stammen nun immer aus dem aktiven Profil, d. h. es findet keine Suche in den Elternprofilen statt (mit „Fix...“ an spezielle Profile gebundene Werte sind hiervon natürlich nicht betroffen)

3.2.2. Interface IFluidAnalysis

Das Interface IFluidAnalysis hat zwei neue Methoden „GetAllFractions“ und „SetAllFractions“, mit denen alle Substanzen gleichzeitig gelesen bzw. geschrieben werden können.

3.2.3. Interface IComp

Eigenschaft „Shape:long“ und read-only Eigenschaft „ShapeCount:long“ zum Lesen und Setzen der aktuellen Darstellungsform bzw. zum Lesen der Anzahl unterschiedlicher Darstellungsformen hinzugefügt.

3.2.4. CoClasses ApplicationBuilder und DllBuilder

Die CoClasses ApplicationBuilder und DllBuilder ermöglichen es, eine EbsOpen-Instanz (Dll-Inproc oder Exe-OutofProcess) schrittweise zu spezifizieren und durch den Aufruf von „Finalize“ zu kreieren. Neu hierbei ist die Möglichkeit, zusätzlich Lizenzoptionen zu spezifizieren.

3.2.5. .net Klasse EbsOpen.Factory

Wie die neuen CoClasses in (3.2.4) unterstützt die Methode „EbsOpen.Factory .Create(InitParams init_params)“ ebenfalls die zusätzlichen Lizenzoptionen. Hierzu verwenden Sie bitte in der Struktur „InitParams“ das Feld „builder_params“ (vom Typ „BuilderParams“).

3.2.6. IApplication / IModel RunExtendedCommand

Mit der Methode „RunExtendedCommand“ kann auf Anwendungs- bzw. auf Modellebene ein benutzerdefinierter Toolbar-Befehl (vgl. 2.5) ausgeführt werden.

3.3. Excel-AddIn

Bei Vorgabe- und Ergebniswerten im Excel-Addin („spec“ und „result“ Typen) können nun auch EbsOpen-Properties der Objekte aufgerufen werden. Z. B. ein Vorgabe- / Ergebniswert „HTX_1. Description“ setzt / liest den Text der „Beschreibung“ im Bauteil namens „HTX_1“.

ACHTUNG: Der Code nach dem Objektnamen wird als C#-Ausdruck ausgewertet, was bei der Angabe weitere Argumente bzw. Weiterverkettung von Properties zu beachten ist.

Weiterhin gibt es nun neben dem EbsScript-Auswertungstypen „expr“ für Ergebnisse einen neuen Typ „assign“ (oder auch „assignment“), mit dem Werte über EbsScript-Syntax vor der Berechnung gesetzt werden können. Hierfür wird dem Ausdruck in der Spalte „Name“ der EbsScript-Zuweisungsoperator „:=“ und der Wert in der Profilspalte angehängt. Wenn in der Einheitenspalte die Einheit „String“ steht, dann wird der Text in Anführungszeichen gesetzt und evtl. darin vorkommende Steuerungszeichen mittels Backslashes geschützt.

Z. B.

Type	Name	Description	Unit	FirstProfile
assign	'@calcoptions.sim.waterSteamTable			1
assign	P4.MEASM		String	P4.MEASM + 0.01

(Bitte beachten Sie das '-Zeichen am Anfang des oberen Ausdrucks, um Excel mitzuteilen, dass der Zelleninhalt als Text interpretiert werden soll.) In diesem Beispiel wird für die erste Zeile der EbsScript-Ausdruck

@calcoptions.sim.waterSteamTable := 1

und für die zweite Zeile der EbsScript-Ausdruck

P4.MEASM := „P4.MEASM + 0.01“

ausgeführt (d. h. in der zweiten Zeile wird beim Bauteil 46 namens „P4“ beim Spezwert „MEASM“ der Ausdruck „P4.MEASM + 0.01“ gesetzt).

Hinweis: Bitte beachten Sie, dass es sich bei den meisten durch EbsOpen-Properties bzw. EbsScript-Zuweisung schreibbaren Objekt-Eigenschaften, um **nicht-profilabhängige** Werte handelt (wie z. B. „Description“). D. h. wird ein solcher Wert in einem Profil geschrieben, dann hat dieser in allen anderen Profilen den gleichen Wert.

3.4. Xui.dll und Programmable.dll

In der Header-Datei „EbsUser.h“ gibt es eine neue Version des Rechen-Interfaces.

Die Version 10 (im Namespace „v10“, nur für C++ verfügbar) beinhaltet eine CallBack-Funktion zur Berechnung von Stoffeigenschaften. Dies macht das Linken zu wdt.lib und mixture.lib obsolet.

3.5. SRxWebAPI

Die SRxWebAPI kann nun auch HTTPS (TSL/SSL) Verbindungen. Ggfs. benötigte Zertifikate müssen über das Betriebssystem in der Windows-Zertifikatsverwaltung hinzugefügt werden!

4. Änderungen in den Ergebnissen

4.1. Dreiwegeventil (Bauteil 49)

Irrtümlicherweise wurde es bei diesem Bauteil bis Release 15 akzeptiert, wenn der Druck auf der Austrittsleitung vorgegeben war. Entsprechend der Ventilstellung wurde dieser dann auf eine der beiden Eintrittsleitungen übertragen. Diese Art der Vorgabe konnte jedoch nur bei einer der beiden Ventilstellungen funktionieren. Schaltet das Ventil um, würde der Druck auf die andere Eintrittsleitung übertragen, auf der er dann doppelt definiert wäre, während die Druckvorgabe auf der anderen Leitung fehlte. Leider wurde in diesem Fall auch keine Fehlermeldung abgesetzt. Das Problem wurde dadurch behoben, dass eine Vorgabe des Drucks auf der Austrittsleitung jetzt nicht mehr akzeptiert wird. Die betroffenen Schaltungen (die ja ohnehin nur bei einer Ventilstellung korrekt funktioniert haben) müssen entsprechend umgebaut werden.

Ebenfalls geändert wurde das Verhalten, wenn alle Massenströme nahezu 0 waren. Hier wurde bisher einfach ein Mittelwert aus beiden Eingängen gebildet. Für die nicht durchströmten Leitungen selbst entsteht durch diese Vereinfachung kein Fehler. Weil der so gebildete Druck jedoch an andere Leitungen weitergegeben werden kann, wurde diese Vereinfachung wieder entfernt.

4.2. Schichtenspeicher (Bauteil 145)

Bis Release 15 war die Berechnung des Fluiddruckes nicht plausibel. In Release 16 wurde die Druckberechnung korrigiert. Daher ändern sich die Ergebniswerte, vor Allem die gerechneten Fluid-Druckwerte an den an das Bauteil angeschlossenen Leitungen sowie der Ergebniswert DPGEOD im Bauteil 145.

Es kann daher nötig sein, die bestehenden Modelle an die Änderung der Release 16 anzupassen, um neu entstandene Fehlermeldungen, wie z. B. Dampfanteile im Wasser, zu eliminieren.

Der Einfluss der Druckberechnung auf die Temperaturfelder ist sehr gering, so dass an dieser Stelle keine signifikante Änderung der Ergebnisse zu erwarten ist.

4.3. k-Werte bei Wärmetauschern

4.3.1. Berücksichtigung beider Wärmeübergangskoeffizienten

Bei der Berechnung des k-Wertes werden jetzt grundsätzlich beide Wärmeübergangskoeffizienten berücksichtigt:

$1/k = 1/al12 + 1/al34$

Bisher wurde beim Eco und beim Verdampfer häufig der al12-Term vernachlässigt, da er im Fall al12 >> al34 nur geringen Einfluss hat. Allerdings erhält man auch bei al12 = 5000 W/m²K und al34 = 50 W/m²K einen Fehler von etwa 1%. Dieser wird ab Release 16 jetzt nicht mehr gemacht.

Allerdings hatte diese Näherung auch bisher keine große Auswirkung, da die Thermodynamik nur vom Produkt k*A abhängt, welches auch bisher schon exakt berechnet wurde. Eine Ungenauigkeit in

A hat sich deshalb im Auslegungsfall nur auf den Ergebniswert A (Fläche) ausgewirkt, nicht aber auf die thermodynamischen Daten, und insbesondere auch nicht auf die Leitungsergebnisse.

Im Hinblick auf Anwendungen, die die Fläche benötigen, oder Arbeitsfluide mit anderen Alpha-Zahlen verwenden, ist die genauere Berechnung sicherlich sinnvoll.

Wenn die genauere Berechnung unerwünscht ist, kann man durch die Vorgabe sehr hoher Zahlenwerte für AL12 praktisch eine Vernachlässigung des AL12-Terms bewirken und die alten Ergebnis wiederherstellen.

In Off-Design kann es durch die Berücksichtigung beider Wärmeübergangskoeffizienten jedoch bei den Bauteilen, bei denen das Teillastverhalten durch Exponenten bestimmt wird, auch zu leichten Änderungen in den Leitungswerten kommen. Ursache hierfür ist die Nichtlinearität, die durch diese Exponenten in die Rechnung hineinkommt, da die Exponenten nicht für k*A als Ganzes, sondern für AL12 und AL34 definiert sind.

4.3.2. Sonderbehandlung Standardwert AL12N = 500 W/m²K

Die Berücksichtigung beider Wärmeübergangskoeffizienten führt allerdings nur dann zu besseren, d. h. realitätsnäheren Ergebnissen, wenn die Wärmeübergangskoeffizienten auch korrekt sind. Wenn jedoch beim Eco oder beim Verdampfer der Standardwert AL12N = 500 W/m²K verwendet wird, werden die Ergebnisse unrealistisch. Leider waren diese 500 W/m²K bis Release 14 auch bei Bauteil 70 (Verdampfer mit Trommel) der Standardwert.

Um zu vermeiden, dass die Berücksichtigung eines AL12N von 500 W/m²K jetzt zu unrealistischen Ergebnissen führt, wurde für diesen Wert eine Sonderbehandlung implementiert: es wird eine Warnung ausgegeben und die Berechnung mit AL12N = 6000 W/m²K (Eco) bzw. AL12N = 10000 W/m²K (Verdampfer) durchgeführt.

4.4. Ergebniswerte in den Bauteilen 128, 129 bei FFU = 0

In den Bauteilen 128 (Steinkohle) und 129 (Braunkohlemühle) nehmen ab Release 16 die Ergebniswerte bei FFU = 0 (Mühle AUS) plausiblere Werte an. Die irrelevanten Ergebniswerte werden auf leer gesetzt.

4.5. Bauteil 131 bei FDELAY > 0

Im Bauteil 131 (instationärer Trenner) wurde ein Fehler behoben, der in Kombination mit dem Schalter FDELAY > 0 aufgetreten war.

4.6. Bauteil 118 bei M1 = M2

Im Bauteil 118 (direkter Speicher) wurde ein Fehler behoben, der im Fall M1 = M2 (Massenströme am Ein- und Austritt sind gleich) aufgetreten war.

5. Bekannte Fehler

5.1. Dokumentation

Leider konnten die neuen Features der Release 16 noch nicht vollständig in die Hilfe aufgenommen werden. Bis ein Patch für die Hilfe verfügbar ist, greifen Sie bitte auf diese Release-Notes zurück.

Es besteht auch die Möglichkeit, auf den im Internet verfügbaren aktuellsten Stand der Online-Hilfe zuzugreifen, in dem man die Hilfeeinstellungen auf „im Browser starten“ umstellt.

STEAG Energy Services GmbH
Wetzbach 35
64673 Zwingenberg
Phone: +49 6251 1059-0
info@ebsilon.com
www.ebsilon.com