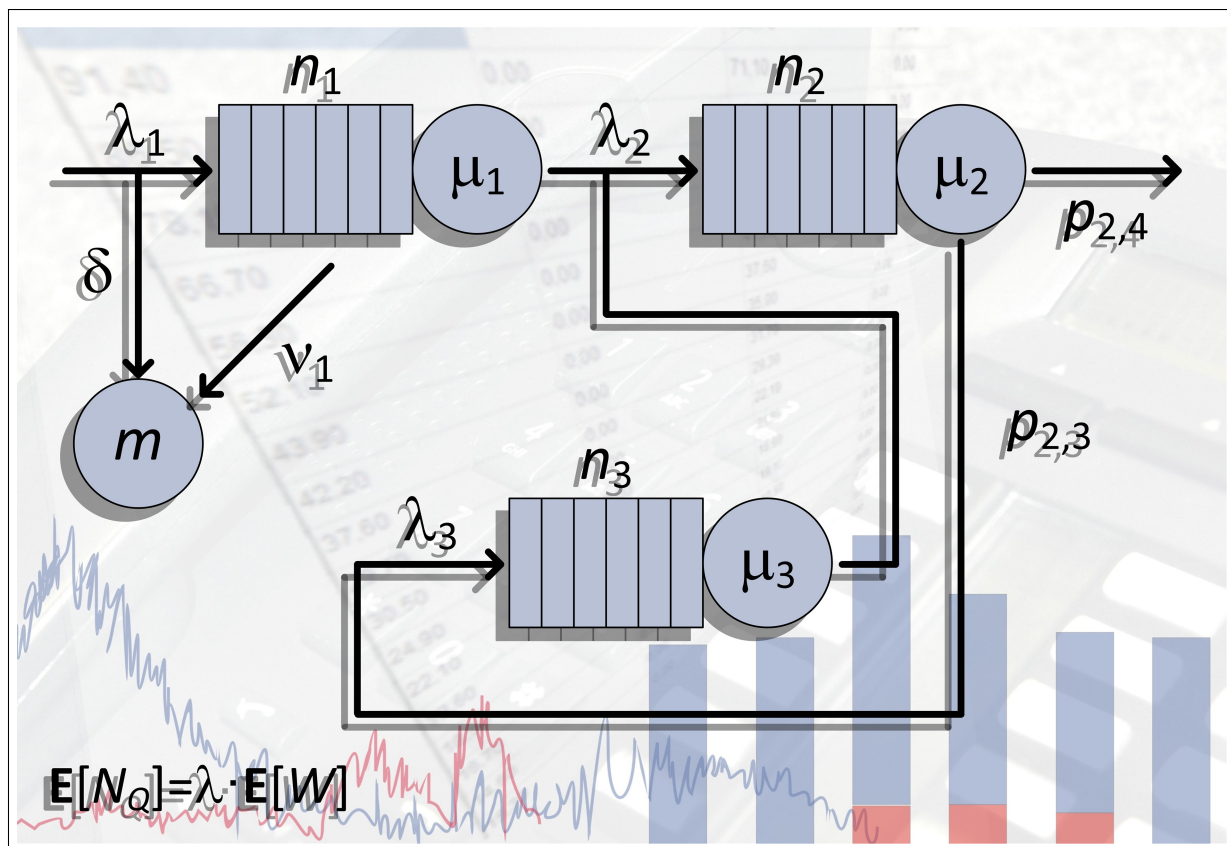


Rechen- und Skriptbefehlsreferenz für den Warteschlangensimulator

ALEXANDER HERZOG (alexander.herzog@tu-clausthal.de)



Diese Referenz bezieht sich auf die Version 4.9.0 des Warteschlangensimulators.
Download-Adresse: <https://github.com/A-Herzog/Warteschlangensimulator/>.

Inhaltsverzeichnis

1	Rechenbefehle und Scripting im Warteschlangensimulator	1
1.1	Ausdrücke erstellen	1
I	Referenz der Rechenbefehle	
2	Konstanten	5
3	Variablen	7
4	Grundrechenarten	9
5	Nachgestellte Anweisungen	11
6	Allgemeine Funktionen	13
6.1	Zufallszahlen	13
7	Winkelfunktionen	15
7.1	Elementare Winkelfunktionen	15
7.2	Hyperbolische Winkelfunktionen	15
7.3	Umkehrfunktionen der elementaren Winkelfunktionen	15
7.4	Umkehrfunktionen der hyperbolischen Winkelfunktionen	15
8	Funktionen mit mehreren Parametern	17
9	Wahrscheinlichkeitsverteilungen	19
9.1	Hypergeometrische Verteilung $Hg(N, K, n)$	19
9.2	Binomial-Verteilung $B(n, p)$	19
9.3	Poisson-Verteilung $P(l)$	19
9.4	Exponentialverteilung mit Mittelwert a	19
9.5	Gleichverteilung über das Intervall $[a; b]$	20

9.6	Normalverteilung mit Mittelwert a und Standardabweichung b	20
9.7	Lognormalverteilung mit Mittelwert a und Standardabweichung b	20
9.8	Gamma-Verteilung mit Parametern $\alpha = a$ und $\beta = b$	20
9.9	Gamma-Verteilung mit Mittelwert a und Standardabweichung b	20
9.10	Erlang-Verteilung mit Parametern n und $\lambda = l$	21
9.11	Beta-Verteilung in dem Intervall $[a; b]$ und mit Parametern $\alpha = c$ und $\beta = d$	21
9.12	Beta-Verteilung in dem Intervall $[a; b]$ und mit Mittelwert c und Standardabweichung d ..	21
9.13	Weibull-Verteilung mit Parametern Scale= a und Form= b	21
9.14	Cauchy-Verteilung mit Mittelwert a und Scale= b	21
9.15	Chi ² -Verteilung mit n Freiheitsgraden	22
9.16	Chi-Verteilung mit n Freiheitsgraden	22
9.17	F-Verteilung mit a Freiheitsgraden im Zähler und b Freiheitsgraden im Nenner	22
9.18	Johnson-SU-Verteilung mit den Parametern $\gamma = a$, $\xi = b$, $\delta = c$ und $\lambda = d$	22
9.19	Dreiecksverteilung über $[a; c]$ mit der höchsten Wahrscheinlichkeitsdichte bei b	22
9.20	Pert-Verteilung über $[a; c]$ mit der höchsten Wahrscheinlichkeitsdichte bei b	23
9.21	Laplace-Verteilung mit Mittelwert mu und Skalierungsfaktor b	23
9.22	Pareto-Verteilung mit Skalierungsparameter $x_{\min} = xmin$ und Formparameter $\alpha = a$	23
9.23	Logistische Verteilung mit Mittelwert $\mu = mu$ und Skalierungsparameter s	23
9.24	Inverse Gauß-Verteilung mit $\lambda = l$ und Mittelwert mu	23
9.25	Rayleigh-Verteilung mit Mittelwert mu	24
9.26	Log-Logistische Verteilung mit α und Mittelwert β	24
9.27	Potenzverteilung auf dem Bereich $[a; b]$ mit Exponent c	24
9.28	Gumbel-Verteilung mit Erwartungswert a und Standardabweichung b	24
9.29	Fatigue-Life-Verteilung mit Lageparameter μ , Skalierungsparameter β und Formparameter γ	24
9.30	Frechet-Verteilung mit Lageparameter δ , Skalierungsparameter β und Formparameter α ..	25
9.31	Hyperbolische Sekanten-Verteilung mit Mittelwert a und Standardabweichung b	25
9.32	Linke Sägezahnverteilung über $[a; b]$	25
9.33	Linke Sägezahnverteilung über mit Erwartungswert a und Standardabweichung b	25
9.34	Rechte Sägezahnverteilung über $[a; b]$	25
9.35	Rechte Sägezahnverteilung über mit Erwartungswert a und Standardabweichung b	26
9.36	Verteilung aus empirischen Daten	26
10	Erlang-C-Rechner	27
11	Allen-Cunneen-Approximationsformel	29
12	Zugriff auf die Modelleigenschaften	31

12.1 Allgemeine Simulationsdaten	31
12.2 Kunden im System	31
12.2.1 Anzahl an Kunden im System	31
12.2.2 Anzahl an wartenden Kunden im System	32
12.3 Kunden an den Stationen	33
12.3.1 Anzahl an Kunden an einer Station	33
12.3.2 Anzahl an Kunden in der Warteschlange an einer Station	34
12.3.3 Anzahl an Kunden in Bedienung an einer Station	35
12.3.4 Anzahl an Ankünften und Abhängen an einer Station	35
12.4 Kunden nach Kundentypen	35
12.4.1 Anzahl an Kunden im System nach Kundentypen	35
12.4.2 Anzahl an wartenden Kunden im System nach Kundentypen	36
12.5 Zähler und Durchsatz	37
12.6 Wartezeiten	37
12.6.1 Wartezeiten an einer Station	37
12.6.2 Wartezeiten über alle Kundentypen	38
12.6.3 Wartezeiten für einen Kundentypen	39
12.7 Transferzeiten	39
12.7.1 Transferzeiten an einer Station	39
12.7.2 Transferzeiten über alle Kundentypen	40
12.7.3 Transferzeiten für einen Kundentypen	41
12.8 Bedienzeiten	41
12.8.1 Bedienzeiten an einer Station	41
12.8.2 Bedienzeiten über alle Kundentypen	42
12.8.3 Bedienzeiten für einen Kundentypen	43
12.9 Verweilzeiten	43
12.9.1 Verweilzeiten an einer Station	43
12.9.2 Verweilzeiten über alle Kundentypen	44
12.9.3 Verweilzeiten für einen Kundentypen	45
12.10 Auslastung der Ressourcen	45
12.10.1 Auslastung einer Ressource	45
12.10.2 Auslastung aller Ressourcen zusammen	46
12.11 Auslastung der Transporter	47
12.11.1 Auslastung einer Transportergruppe	47
12.11.2 Auslastung aller Transporter zusammen	48
12.12 Zugriff auf Statistik-Stationen Datenfelder	48
12.13 Zugriff auf Analogwerte	49

12.14	Zugriff auf Kundenobjekt-spezifische Datenfelder	49
12.15	Zugriff auf die Kosten	50
13	Vergleiche	53
13.1	Vergleichsfunktion	53
II	Referenz der Javascript-Befehle	
14	Statistics-Objekt	57
14.1	Definition des Ausgabeformats	57
14.2	Zugriff auf die Statistik-XML-Daten	58
14.3	Speichern der Statistikdaten in Dateien	59
14.4	Zugriff auf das Modell	59
14.5	Abfrage der zugehörigen Statistikdatei	59
15	System-Objekt	61
16	Simulation-Objekt	63
16.1	Basisfunktionen	63
16.2	Zugriff auf kundenspezifische Daten	64
16.3	Temporäre Batche	66
16.4	Zugriff auf Parameter des Simulationsmodells	67
16.5	Zugriff auf den aktuellen Eingabewert	67
16.6	Anzahl an Bedienern in einer Ressource	67
16.7	Signale auslösen	68
16.8	Meldung in Logging ausgeben	68
16.9	Kunden an Verzögerungen-Stationen freigeben	68
17	Clients-Objekt	69
18	Output-Objekt	71
19	FileOutput-Objekt	73
20	Model-Objekt	75
21	XML-Auswahlbefehle	77
III	Referenz der Java-Befehle	
22	StatisticsInterface abrufbar über <code>sim.getStatistics()</code>	81

22.1	Definition des Ausgabeformats	81
22.2	Zugriff auf die Statistik-XML-Daten	82
22.3	Speichern der Statistikdaten in Dateien	83
22.4	Zugriff auf das Modell.....	83
22.5	Abfrage der zugehörigen Statistikdatei	83
23	RuntimeInterface abrufbar über <code>sim.getRuntime</code>	85
24	SystemInterface abrufbar über <code>sim.getSystem()</code>	87
24.1	Basisfunktionen	87
24.2	Zugriff auf Parameter des Simulationsmodells	87
24.3	Anzahl an Bedienern in einer Ressource	88
24.4	Signale auslösen	89
24.5	Externen Code aufrufen	89
24.6	Meldung in Logging ausgeben	89
24.7	Kunden an Verzögerungen-Stationen freigeben	89
25	ClientInterface abrufbar über <code>sim.getClient()</code>	91
25.1	Temporäre Batche	92
26	InputValueInterface abrufbar über <code>sim.getInputValue()</code>	95
27	ClientsInterface abrufbar über <code>sim.getClients()</code>	97
28	OutputInterface abrufbar über <code>sim.getOutput()</code>	99
29	FileOutputInterface abrufbar über <code>sim.getFileOutput()</code>	101
30	ModelInterface abrufbar über <code>sim.getModel()</code>	103
31	XML-Auswahlbefehle	105


Kapitel 1

Rechenbefehle und Scripting im Warteschlangensimulator

Rechenbefehle können im Simulator verwendet werden, um z.B. Zeitdauern (wie Bedienzeiten) zu bestimmen oder um festzulegen, in welche Verzweigungsrichtung ein Kunde geleitet werden soll.

Sowohl zur Bestimmung von Verzweigungsrichtungen als auch bei der Auswertung von Simulationsergebnissen und bei der Durchführung von Parameterreihen können **Skripte** eingesetzt werden. Der Warteschlangensimulator verwendet dabei als Sprachen Javascript und Java.

1.1 Ausdrücke erstellen

Rechts neben allen Eingabefeldern, in die ein Rechenbefehl eingegeben werden kann, ist stets eine Schaltfläche mit dem folgenden Symbol zu sehen:  Über diese Schaltfläche kann der **Ausdruck bearbeiten** Dialog (siehe Abbildung 1.1) aufgerufen werden. Der Dialog enthält eine vollständige Liste aller im aktuellen Kontext zur Verfügung stehenden Befehle und ermöglicht das einfache Zusammenstellen komplexerer Befehle und Ausdrücke.

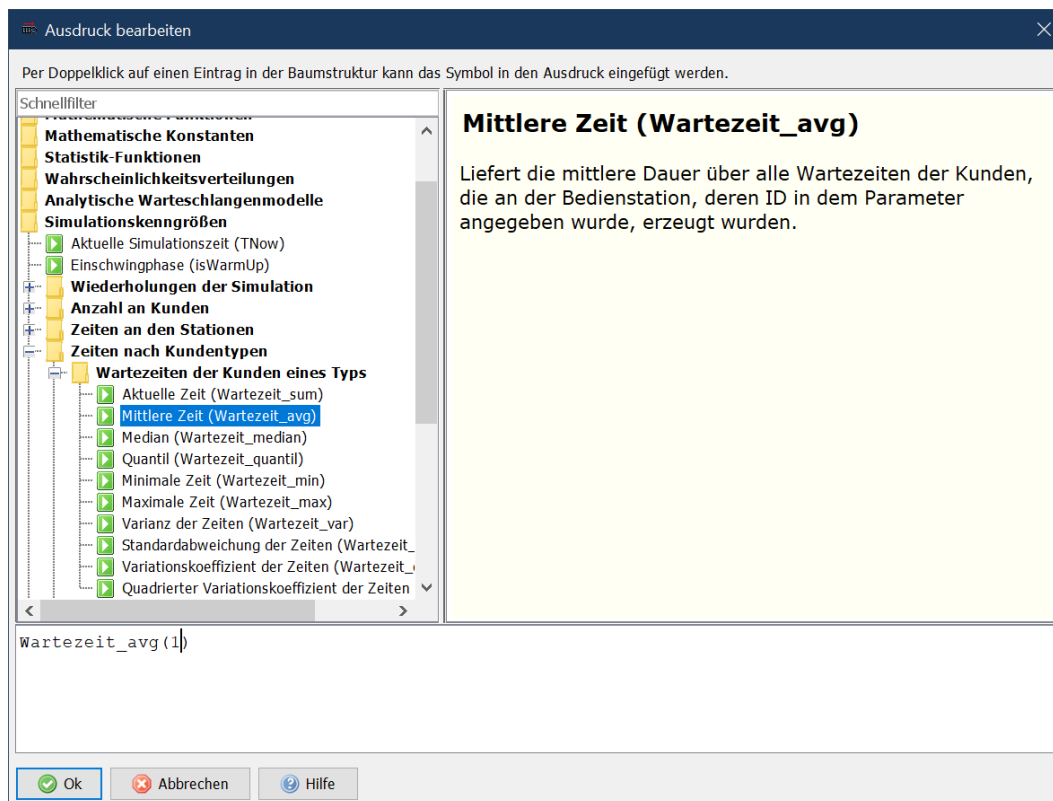


Abb. 1.1: „Ausdruck bearbeiten“-Dialog

Teil I

Referenz der Rechenbefehle

Bei der Verwendung von Rechenbefehlen im Warteschlangensimulator ist zwischen **Ausdrücken** und **Vergleichen** zu unterscheiden. Ausdrücke dienen dazu, einen Zahlenwert zu berechnen, der dann z.B. als Zeitdauer verwendet wird. Vergleiche liefern eine ja/nein Entscheidung (z.B. ob ein Kunde in eine bestimmte Richtung geleitet werden soll). Im Gegensatz zu Ausdrücken besitzen Vergleiche immer mindestens einen Vergleichsoperator.

Alle im Folgenden vorgestellten Befehle werden jeweils in jeder beliebigen **Groß- und Kleinschreibung** erkannt, d.h. es wird nicht zwischen verschiedenen Groß-/Kleinschreibweisen unterschieden.

Kapitel 2

Konstanten

Folgende Konstanten stehen in den allen Rechenbefehlen zur Verfügung:

- „e”: Liefert die Basis der Exponentialfunktion e^x . Es gilt $e \approx 2,718281828459$.
- „pi”: Liefert den Wert der Kreiskonstante π . Es gilt $\pi \approx 3,1415926535898$.

Kapitel 3

Variablen

Werden Ausdrücke im Kontext eines Kunden berechnet, so stehen die Variablen

- „w” für die **bisherige Wartezeit** des Kunden,
- „t” für die **bisherige Transferzeit** des Kunden und
- „p” für die **bisherige Bedienzeit** des Kunden zur Verfügung.

Bei der Berechnung von Score-Werten wird „w” abweichend nicht mit der bisherigen gesamten Wartezeit des Kunden belegt, sondern mit der bisherigen Wartezeit an der aktuellen Station.

Des weiteren stehen stets alle Variablen, die über ein Zuweisungselement definiert werden, zur Verfügung. Vor der ersten Zuweisung eines Wertes an eine Variable hat diese den Wert 0.

Kapitel 4

Grundrechenarten

Es werden die üblichen Grundrechenarten unterstützt:

- Addition: „+“
- Subtraktion: „-“
- Multiplikation: „*“
- Division: „/“
- Potenzieren: „^“

Die Regel **Punkt- vor Strichrechnung** wird dabei berücksichtigt. Um davon abweichende Auswertungen zu erzwingen, können **Klammern** gesetzt werden.

Kapitel 5

Nachgestellte Anweisungen

Folgende Ausdrücke können unmittelbar hinter eine Zahl geschrieben werden:

- „%“: Der Zahlenwert vor diesem Symbol wird als Prozentwert interpretiert, z.B. $30\% = 0,3$.
- „²“: Potenziert die Zahl vor diesem Symbol mit 2.
- „³“: Potenziert die Zahl vor diesem Symbol mit 3.
- „!“: Berechnet die Fakultät der Zahl vor diesem Ausdruck, z.B. $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$.
- „°“: Rechnet die Zahl vor diesem Symbol von Grad nach Bogenmaß um, z.B. $180^\circ = 3,1415 \dots$
(Siehe hierzu auch Abschnitt 7 in dem die vom Warteschlangensimulator unterstützten Winkelfunktionen vorgestellt werden.)

Kapitel 6

Allgemeine Funktionen

- „**abs(x)**“: Absolutbetrag, z.B. **abs(-5)=5**.
- „**ceil(x)**“: Aufrunden, z.B. **ceil(2,1)=3**
- „**exp(x)**“: Exponentialfunktion e^x .
- „**factorial(x)**“: Fakultät, z.B. $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$.
- „**binom(n;k)**“: Binomialkoeffizient
- „**floor(x)**“: Abrunden, z.B. **floor(2,9)=2**
- „**frac(x)**“: Nachkommaanteil, z.B. **frac(1,3)=0,3**
- „**gamma(x)**“: Gamma-Funktion, z.B. **gamma(5)=4!=24**
- „**int(x)**“: Ganzzahlanteil, z.B. **int(2,9)=2**
- „**log(x)**“: Logarithmus zur Basis e .
- „**log(x;b)**“: Logarithmus zur Basis b .
- „**ld(x)**“: Logarithmus zur Basis 2, z.B. **ld(256)=8**.
- „**lg(x)**“: Logarithmus zur Basis 10, z.B. **lg(100)=2**.
- „**ln(x)**“: Logarithmus zur Basis e .
- „**modulo(a;b)**“ oder „**mod(a;b)**“: Divisionsrest bei Division a/b
- „**pow(x;y)**“: Potenzieren x^y .
- „**round(x)**“: Runden, z.B. **round(4,4)=4** und **round(4,5)=5**.
- „**sign(x)**“: Vorzeichen einer Zahl, z.B. **sign(3)=1** und **sign(-3)=-1**.
- „**sqrt(x)**“: Quadratwurzel, z.B. **sqrt(81)=9**.
- „**sqr(x)**“: Zahl quadrieren, z.B. **sqr(4)=16**.

6.1 Zufallszahlen

Mit den folgenden Befehlen können Zufallszahlen, die in einem bestimmten Bereich **gleichverteilt** sind erzeugt werden. Im Abschnitt 9 werden weitere Funktionen zur Erzeugung von Zufallszahlen gemäß bestimmten Verteilungsfunktionen vorgestellt.

- „**random()**“: Liefert eine Zufallszahl zwischen 0 (einschließlich) und 1 (ausschließlich).

- „**random(x)**“: Liefert eine Zufallszahl zwischen 0 (einschließlich) und x (ausschließlich).

Kapitel 7

Winkelfunktionen

Die Winkelfunktionen beziehen sich immer auf 2π als Vollkreis (Bogenmaß, „Rad“). Wenn Winkel in Grad (360° für den Vollkreis) angegeben werden sollen, so müssen diese bei der Verwendung der elementaren Winkelfunktionen in Bogenmaß umgerechnet werden, z.B. **$\sin(90^\circ)=1$** .

7.1 Elementare Winkelfunktionen

- „ **$\sin(x)$** “: Sinus
- „ **$\cos(x)$** “: Cosinus
- „ **$\tan(x)$** “: Tangens
- „ **$\cot(x)$** “: Cotangens

7.2 Hyperbolische Winkelfunktionen

- „ **$\sinh(x)$** “: Sinus hyperbolicus
- „ **$\cosh(x)$** “: Cosinus hyperbolicus
- „ **$\tanh(x)$** “: Tangens hyperbolicus
- „ **$\coth(x)$** “: Cotangens hyperbolicus

7.3 Umkehrfunktionen der elementaren Winkelfunktionen

- „ **$\arcsin(x)$** “: Arcus-Sinus
- „ **$\arccos(x)$** “: Arcus-Cosinus
- „ **$\arctan(x)$** “: Arcus-Tangens
- „ **$\text{arccot}(x)$** “: Arcus-Cotangens

7.4 Umkehrfunktionen der hyperbolischen Winkelfunktionen

- „ **$\text{arcsinh}(x)$** “: Arcus-Sinus hyperbolicus

- „ $\operatorname{arccosh}(x)$ “: Arcus-Cosinus hyperbolicus
- „ $\operatorname{arctanh}(x)$ “: Arcus-Tangens hyperbolicus
- „ $\operatorname{arccoth}(x)$ “: Arcus-Cotangens hyperbolicus

Kapitel 8

Funktionen mit mehreren Parametern

Die folgenden Funktionen können beliebig viele Parameter entgegennehmen. Die einzelnen Parameter müssen dabei durch Semikolon „;“ getrennt angegeben werden.

- „**Min(a;b;c;...)**“: Berechnet das Minimum der übergebenen Zahlen.
- „**Max(a;b;c;...)**“: Berechnet das Maximum der übergebenen Zahlen.
- „**Sum(a;b;c;...)**“: Berechnet die Summe der übergebenen Zahlen.
- „**Mean(a;b;c;...)**“: Berechnet das arithmetische Mittel der übergebenen Zahlen.
- „**Median(a;b;c;...)**“: Berechnet den Median der übergebenen Zahlen.
- „**Var(a;b;c;...)**“: Berechnet die korrigierte Stichprobenvarianz der übergebenen Zahlen.
- „**SD(a;b;c;...)**“: Berechnet die korrigierte Stichprobenstandardabweichung der übergebenen Zahlen.
- „**SCV(a;b;c;...)**“: Berechnet den quadrierten Variationskoeffizient der übergebenen Zahlen.
- „**CV(a;b;c;...)**“: Berechnet den Variationskoeffizient der übergebenen Zahlen.

Kapitel 9

Wahrscheinlichkeitsverteilungen

Mit Hilfe der folgenden Befehle können sowohl Werte der Dichte und der Verteilungsfunktion der folgenden Wahrscheinlichkeitsverteilungen berechnet werden als auch Zufallszahlen auf Basis einer dieser Wahrscheinlichkeitsverteilungen berechnet werden:

9.1 Hypergeometrische Verteilung $Hg(N, K, n)$

- „`HypergeometricDist(k;N;K;n)`“: Berechnet die Zähldichte an der Stelle k .
- „`HypergeometricDist(N;K;n)`“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.2 Binomial-Verteilung $B(n, p)$

- „`BinomialDist(k;n;p)`“: Berechnet die Zähldichte an der Stelle k .
- „`BinomialDist(n;p)`“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.3 Poisson-Verteilung $P(l)$

- „`PoissonDist(k;1)`“: Berechnet die Zähldichte an der Stelle k .
- „`PoissonDist(1)`“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.4 Exponentialverteilung mit Mittelwert a

- „`ExpDist(x;a;0)`“: Berechnet die Dichte an der Stelle x .
- „`ExpDist(x;a;1)`“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „`ExpDist(a)`“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „`ExpDistRange(min;max;a)`“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.5 Gleichverteilung über das Intervall $[a; b]$

- „UniformDist(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „UniformDist(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „UniformDist(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.6 Normalverteilung mit Mittelwert a und Standardabweichung b

- „NormalDist(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „NormalDist(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „NormalDist(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „NormalDistRange(min;max;a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.7 Lognormalverteilung mit Mittelwert a und Standardabweichung b

- „LogNormalDist(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „LogNormalDist(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „LogNormalDist(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „LogNormalDistRange(min;max;a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.8 Gamma-Verteilung mit Parametern $\alpha = a$ und $\beta = b$

- „GammaDist(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „GammaDist(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „GammaDist(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „GammaDistRange(min;max;a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.9 Gamma-Verteilung mit Mittelwert a und Standardabweichung b

- „GammaDistDirect(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „GammaDistDirect(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „GammaDistDirect(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „GammaDistDirectRange(min;max;a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.10 Erlang-Verteilung mit Parametern n und $\lambda = l$

- „ErlangDist($x;n;l;0$)“: Berechnet die Dichte an der Stelle x .
- „ErlangDist($x;n;l;1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „ErlangDist($n;b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „ErlangDistRange($min;max;n;b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.11 Beta-Verteilung in dem Intervall $[a; b]$ und mit Parametern $\alpha = c$ und $\beta = d$

- „BetaDist($x;a;b;c;d;0$)“: Berechnet die Dichte an der Stelle x .
- „BetaDist($x;a;b;c;d;1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „BetaDist($a;b;c;d$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.12 Beta-Verteilung in dem Intervall $[a; b]$ und mit Mittelwert c und Standardabweichung d

- „BetaDistDirect($x;a;b;c;d;0$)“: Berechnet die Dichte an der Stelle x .
- „BetaDistDirect($x;a;b;c;d;1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „BetaDistDirect($a;b;c;d$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.13 Weibull-Verteilung mit Parametern Scale= a und Form= b

- „WeibullDist($x;a;b;0$)“: Berechnet die Dichte an der Stelle x .
- „WeibullDist($x;a;b;1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „WeibullDist($a;b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „WeibullDistRange($min;max;a;b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.14 Cauchy-Verteilung mit Mittelwert a und Scale= b

- „CauchyDist($x;a;b;0$)“: Berechnet die Dichte an der Stelle x .
- „CauchyDist($x;a;b;1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „CauchyDist($a;b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „CauchyDistRange($min;max;a;b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.15 Chi²-Verteilung mit n Freiheitsgraden

- „ChiSquareDist($x;n;0$)“: Berechnet die Dichte an der Stelle x .
- „ChiSquareDist($x;n;1$)“: Chi²-Verteilung mit n Freiheitsgraden.
- „ChiSquareDist(n)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „ChiSquareDistRange($min;max;n$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.16 Chi-Verteilung mit n Freiheitsgraden

- „ChiDist($x;n;0$)“: Berechnet die Dichte an der Stelle x .
- „ChiDist($x;n;1$)“: Chi-Verteilung mit n Freiheitsgraden.
- „ChiDist(n)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „ChiDistRange($min;max;n$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.17 F-Verteilung mit a Freiheitsgraden im Zähler und b Freiheitsgraden im Nenner

- „FDist($x;a;b;0$)“: Berechnet die Dichte an der Stelle x .
- „FDist($x;a;b;1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „FDist($a;b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „FDistRange($min;max;a;b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.18 Johnson-SU-Verteilung mit den Parametern $\gamma = a$, $\xi = b$, $\delta = c$ und $\lambda = d$

- „JohnsonSUDist($x;a;b;c;d;0$)“: Berechnet die Dichte an der Stelle x .
- „JohnsonSUDist($x;a;b;c;d;1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „JohnsonSUDist($a;b;c;d$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „JohnsonSUDistRange($min;max;a;b;c;d$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.19 Dreiecksverteilung über $[a; c]$ mit der höchsten Wahrscheinlichkeitsdichte bei b

- „TriangularDist($x;a;b;c;0$)“: Berechnet die Dichte an der Stelle x .
- „TriangularDist($x;a;b;c;1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „TriangularDist($a;b;c$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.20 Pert-Verteilung über $[a; c]$ mit der höchsten Wahrscheinlichkeitsdichte bei b

- „PertDist($x; a; b; c; 0$)“: Berechnet die Dichte an der Stelle x .
- „PertDist($x; a; b; c; 1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „PertDist($a; b; c$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.21 Laplace-Verteilung mit Mittelwert μ und Skalierungsfaktor b

- „LaplaceDist($x; \mu; b; 0$)“: Berechnet die Dichte an der Stelle x .
- „LaplaceDist($x; \mu; b; 1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „LaplaceDist($\mu; b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „LaplaceDistRange($\min; \max; \mu; b$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von \min bis \max befindet.

9.22 Pareto-Verteilung mit Skalierungsparameter $x_{\min} = x_{\min}$ und Formparameter $\alpha = a$

- „ParetoDist($x; x_{\min}; a; 0$)“: Berechnet die Dichte an der Stelle x .
- „ParetoDist($x; x_{\min}; a; 1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „ParetoDist($x_{\min}; a$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.23 Logistische Verteilung mit Mittelwert $\mu = \mu$ und Skalierungsparameter s

- „LogisticDist($x; \mu; s; 0$)“: Berechnet die Dichte an der Stelle x .
- „LogisticDist($x; \mu; s; 1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „LogisticDist($\mu; s$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „LogisticDistRange($\min; \max; \mu; s$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von \min bis \max befindet.

9.24 Inverse Gauß-Verteilung mit $\lambda = l$ und Mittelwert μ

- „InverseGaussianDist($x; l; \mu; 0$)“: Berechnet die Dichte an der Stelle x .
- „InverseGaussianDist($x; l; \mu; 1$)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „InverseGaussianDist($l; \mu$)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „InverseGaussianDistRange($\min; \max; l; \mu$)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von \min bis \max befindet.

9.25 Rayleigh-Verteilung mit Mittelwert μ

- „RayleighDist(x;mu;0)“: Berechnet die Dichte an der Stelle x .
- „RayleighDist(x;mu;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „RayleighDist(mu)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „RayleighDistRange(min;max;mu)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von \min bis \max befindet.

9.26 Log-Logistische Verteilung mit α und Mittelwert β

- „LogLogisticDist(x;alpha;beta;0)“: Berechnet die Dichte an der Stelle x .
- „LogLogisticDist(x;alpha;beta;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „LogLogisticDist(alpha;beta)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „LogLogisticDistRange(min;max;alpha;beta)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von \min bis \max befindet.

9.27 Potenzverteilung auf dem Bereich $[a; b]$ mit Exponent c

- „PowerDist(x;a;b;c;0)“: Berechnet die Dichte an der Stelle x .
- „PowerDist(x;a;b;c;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „PowerDist(a;b;c)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.28 Gumbel-Verteilung mit Erwartungswert a und Standardabweichung b

- „GumbelDist(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „GumbelDist(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „GumbelDist(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „GumbelDistRange(min;max;a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von \min bis \max befindet.

9.29 Fatigue-Life-Verteilung mit Lageparameter μ , Skalierungsparameter β und Formparameter γ

- „FatigueLifeDist(x;mu;beta;gamma;0)“: Berechnet die Dichte an der Stelle x .
- „FatigueLifeDist(x;mu;beta;gamma;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „FatigueLifeDist(mu;beta;gamma)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „FatigueLifeDistRange(min;max;mu;beta;gamma)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von \min bis \max befindet.

9.30 Frechet-Verteilung mit Lageparameter δ , Skalierungsparameter β und Formparameter α

- „FrechetDist(x;delta;beta;alpha;0)“: Berechnet die Dichte an der Stelle x .
- „FrechetDist(x;delta;beta;alpha;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „FrechetDist(delta;beta;alpha)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „FrechetDistRange(min;max;delta;beta;alpha)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.31 Hyperbolische Sekanten-Verteilung mit Mittelwert a und Standardabweichung b

- „HyperbolicSecantDist(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „HyperbolicSecantDist(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „HyperbolicSecantDist(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.
- „HyperbolicSecantDistRange(min;max;a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung und stellt dabei sicher, dass sich das Ergebnis in dem Bereich von min bis max befindet.

9.32 Linke Sägezahnverteilung über $[a; b]$

- „LeftSawtoothDist(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „LeftSawtoothDist(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „LeftSawtoothDist(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.33 Linke Sägezahnverteilung über mit Erwartungswert a und Standardabweichung b

- „LeftSawtoothDistDirect(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „LeftSawtoothDistDirect(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „LeftSawtoothDistDirect(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.34 Rechte Sägezahnverteilung über $[a; b]$

- „RightSawtoothDist(x;a;b;0)“: Berechnet die Dichte an der Stelle x .
- „RightSawtoothDist(x;a;b;1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „RightSawtoothDist(a;b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.35 Rechte Sägezahnverteilung über mit Erwartungswert a und Standardabweichung b

- „RightSawtoothDistDirect(x ; a ; b ; 0)“: Berechnet die Dichte an der Stelle x .
- „RightSawtoothDistDirect(x ; a ; b ; 1)“: Berechnet den Wert der Verteilungsfunktion an der Stelle x .
- „RightSawtoothDistDirect(a ; b)“: Erzeugt eine Zufallszahl gemäß der Verteilung.

9.36 Verteilung aus empirischen Daten

- „EmpirischeDichte(x ; wert1; wert2; wert3; ...; max)“:
Berechnet die Dichte an der Stelle x . Die angegebenen Werte werden dabei auf den Bereich von 0 bis max verteilt.
- „EmpirischeVerteilung(x ; wert1; wert2; wert3; ...; max)“:
Berechnet den Wert der Verteilungsfunktion an der Stelle x . Die angegebenen Werte werden dabei auf den Bereich von 0 bis max verteilt.
- „EmpirischeZufallszahl(wert1; wert2; wert3; ...; max)“:
Erzeugt eine Zufallszahl gemäß der Verteilung. Die angegebenen Werte werden dabei auf den Bereich von 0 bis max verteilt.
- „EmpirischeVerteilungMittelwert(wert1; wert2; wert3; ...; max)“:
Liefert den Erwartungswert der Verteilung.
- „EmpirischeVerteilungMedian(wert1; wert2; wert3; ...; max)“:
Liefert den Median der Verteilung.
- „EmpirischeVerteilungQuantil(wert1; wert2; wert3; ...; max; p)“:
Liefert das Quantil zur Wahrscheinlichkeit p der Verteilung.
- „EmpirischeVerteilungSD(wert1; wert2; wert3; ...; max)“:
Liefert die Standardabweichung der Verteilung.
- „EmpirischeVerteilungVar(wert1; wert2; wert3; ...; max)“:
Liefert die Varianz der Verteilung.
- „EmpirischeVerteilungCV(wert1; wert2; wert3; ...; max)“:
Liefert den Variationskoeffizient der Verteilung.

Kapitel 10

Erlang-C-Rechner

Mit Hilfe des folgenden Befehls können verschiedene Kenngrößen auf Basis der erweiterten Erlang-C-Formel berechnet werden:

- „`ErlangC(lambda;mu;nu;c;K;-1)`“:
Berechnet die mittlere Warteschlangenlänge $\mathbf{E}[N_Q]$.
- „`ErlangC(lambda;mu;nu;c;K;-2)`“:
Berechnet die mittlere Anzahl an Kunden im System $\mathbf{E}[N]$.
- „`ErlangC(lambda;mu;nu;c;K;-3)`“:
Berechnet die mittlere Wartezeit $\mathbf{E}[W]$.
- „`ErlangC(lambda;mu;nu;c;K;-4)`“:
Berechnet die mittlere Verweilzeit $\mathbf{E}[V]$.
- „`ErlangC(lambda;mu;nu;c;K;-5)`“:
Berechnet die mittlere Erreichbarkeit $1 - P(A)$.
- „`ErlangC(lambda;mu;nu;c;K;t)`“:
Berechnet die Wahrscheinlichkeit für den Service-Level an der t -Sekunden-Schranke $P(W \leq t)$.

Die Parameter haben dabei folgende Bedeutungen:

- **lambda:**
Ankunftsrate λ (in Kunden pro Zeiteinheit), d.h. Kehrwert der mittleren Zwischenankunftszeit.
- **mu:**
Bedienrate μ (in Kunden pro Zeiteinheit), d.h. Kehrwert der mittleren Bediendauer.
- **nu:**
Abbruchrate ν (in Kunden pro Zeiteinheit), d.h. Kehrwert der mittleren Wartezeittoleranz.
- **c:**
Anzahl an verfügbaren parallel arbeitenden Bedienern.
- **K:**
Anzahl an verfügbaren Plätzen im System (Warte- und Bedienplätze zusammen, d.h. es gilt $K \geq c$).

Kapitel 11

Allen-Cunneen-Approximationsformel

Mit Hilfe des folgenden Befehls können verschiedene Kenngrößen auf Basis der Allen-Cunneen-Approximationsformel berechnet werden:

- `„AllenCunneen(lambda;mu;cvI;cvS;c;-1)“`:
Berechnet die mittlere Warteschlangenlänge $E[N_Q]$.
- `„AllenCunneen(lambda;mu;cvI;cvS;c;-2)“`:
Berechnet die mittlere Anzahl an Kunden im System $E[N]$.
- `„AllenCunneen(lambda;mu;cvI;cvS;c;-3)“`:
Berechnet die mittlere Wartezeit $E[W]$.
- `„AllenCunneen(lambda;mu;cvI;cvS;c;-4)“`:
Berechnet die mittlere Verweilzeit $E[V]$.

Die Parameter haben dabei folgende Bedeutungen:

- **lambda**:
Ankunftsrate λ (in Kunden pro Zeiteinheit), d.h. Kehrwert der mittleren Zwischenankunftszeit.
- **mu**:
Bedienrate μ (in Kunden pro Zeiteinheit), d.h. Kehrwert der mittleren Bediendauer.
- **cvI**:
Variationskoeffizient der Zwischenankunftszeiten $CV[I]$ (kleine Werte bedeuten, dass die Ankünfte sehr gleichmäßig erfolgen).
- **cvS**:
Variationskoeffizient der Bedienzeiten $CV[S]$ (kleine Werte bedeuten, dass die Bedienung sehr gleichmäßig erfolgen).
- **c**:
Anzahl an verfügbaren parallel arbeitenden Bedienern.

Kapitel 12

Zugriff auf die Modelleigenschaften

12.1 Allgemeine Simulationsdaten

- „**SimTime()**” oder „**TNow()**”:
Liefert die aktuelle Simulationszeit in Sekunden.
- „**WarmUp()**” oder „**isWarmUp()**”:
Liefert 1 zurück, wenn sich die Simulation noch in der Einschwingphase befindet, sonst 0.
- „**WiederholungNummer()**” oder „**RepeatCurrent()**”:
Liefert die aktuell in Bearbeitung befindliche Wiederholung der Simulation (1-basierender Wert).
- „**WiederholungenAnzahl()**” oder „**RepeatCount()**”:
Liefert die geplante Anzahl an Wiederholungen der Simulation.
- „**\$(“Name”)**”:
Liefert die ID des Elements mit dem Namen, der innerhalb der Anführungszeichen steht. Existiert keine Station mit dem angegebenen Namen, so liefert die Funktion -1.
- „**\$(“Schlüssel”)**”:
Liefert den Wert eines über **getMapGlobal()** in einem Scripting-Element eingestellten Zuordnungswertes.

12.2 Kunden im System

12.2.1 Anzahl an Kunden im System

- „**WIP()**” oder „**N()**” oder „**Station()**”:
Liefert die aktuelle Gesamtanzahl an Kunden im System.
- „**WIP_avg()**” oder „**Station_avg()**” oder „**N_avg()**” oder „**WIP_Mittelwert()**” oder „**Station_Mittelwert()**” oder „**N_Mittelwert()**”:
Liefert die mittlere Anzahl an Kunden im System.
- „**WIP_median()**” oder „**Station_median()**” oder „**N_median()**”:
Liefert den Median der Anzahl an Kunden im System.
- „**WIP_quantil(p)**” oder „**Station_quantil(p)**” oder „**N_quantil(p)**”:
Liefert das Quantil zur Wahrscheinlichkeit p der Anzahl an Kunden im System.

- „WIP_min()“ oder „Station_min()“ oder „N_min()“ oder „WIP_Minimum()“ oder „Station_Minimum()“ oder „N_Minimum()“:
Liefert die minimale Anzahl an Kunden im System.
- „WIP_max()“ oder „Station_max()“ oder „N_max()“ oder „WIP_Maximum()“ oder „Station_Maximum()“ oder „N_Maximum()“:
Liefert die maximale Anzahl an Kunden im System.
- „WIP_var()“ oder „Station_var()“ oder „N_var()“ oder „WIP_Varianz()“ oder „Station_Varianz()“ oder „N_Varianz()“:
Liefert die Varianz der Anzahl an Kunden im System.
- „WIP_sd()“ oder „Station_sd()“ oder „N_sd()“ oder „WIP_Standardabweichung()“ oder „Station_Standardabweichung()“ oder „N_Standardabweichung()“:
Liefert die Standardabweichung der Anzahl an Kunden im System.
- „WIP_cv()“ oder „Station_cv()“ oder „N_cv()“:
Liefert den Variationskoeffizienten der Anzahl an Kunden im System.
- „WIP_scv()“ oder „Station_scv()“ oder „N_scv()“:
Liefert den quadrierten Variationskoeffizienten der Anzahl an Kunden im System.

12.2.2 Anzahl an wartenden Kunden im System

- „NQ()“ oder „Queue()“ oder „Schlange()“ oder „Warteschlange()“:
Liefert die aktuelle Anzahl an wartenden Kunden im System.
- „NQ_avg()“ oder „Queue_avg()“ oder „Schlange_avg()“ oder „Warteschlange_avg()“ oder „NQ_Mittelwert()“ oder „Queue_Mittelwert()“ oder „Schlange_Mittelwert()“ oder „Warteschlange_Mittelwert()“:
Liefert die mittlere Anzahl an wartenden Kunden im System.
- „NQ_median()“ oder „Queue_median()“ oder „Schlange_median()“ oder „Warteschlange_median()“:
Liefert den Median der Anzahl an Kunden in allen Warteschlange zusammen.
- „NQ_quantil(p)“ oder „Queue_quantil(p)“ oder „Schlange_quantil(p)“ oder „Warteschlange_quantil(p)“:
Liefert das Quantil zur Wahrscheinlichkeit p der Anzahl an Kunden in allen Warteschlange zusammen.
- „NQ_min()“ oder „Queue_min()“ oder „Schlange_min()“ oder „Warteschlange_min()“ oder „NQ_Minimum()“ oder „Queue_Minimum()“ oder „Schlange_Minimum()“ oder „Warteschlange_Minimum()“:
Liefert die minimale Anzahl an wartenden Kunden im System.
- „NQ_max()“ oder „Queue_max()“ oder „Schlange_max()“ oder „Warteschlange_max()“ oder „NQ_Maximum()“ oder „Queue_Maximum()“ oder „Schlange_Maximum()“ oder „Warteschlange_Maximum()“:
Liefert die maximale Anzahl an wartenden Kunden im System.
- „NQ_var()“ oder „Queue_var()“ oder „Schlange_var()“ oder „Warteschlange_var()“ oder „NQ_Varianz()“ oder „Queue_Varianz()“ oder „Schlange_Varianz()“ oder „Warteschlange_Varianz()“:
Liefert die Varianz der Anzahl an wartenden Kunden im System.
- „NQ_sd()“ oder „Queue_sd()“ oder „Schlange_sd()“ oder „Warteschlange_sd()“ oder „NQ_Standardabweichung()“ oder „Queue_Standardabweichung()“ oder „Schlange_Standardabweichung()“ oder „Warteschlange_Standardabweichung()“:
Liefert die Standardabweichung der Anzahl an wartenden Kunden im System.

„Schlange_Standardabweichung()“ oder „Warteschlange_Standardabweichung()“:
Liefert die Standardabweichung der Anzahl an wartenden Kunden im System.

- „NQ_cv()“ oder „Queue_cv()“ oder „Schlange_cv()“ oder „Warteschlange_cv()“:
Liefert den Variationskoeffizienten der Anzahl an wartenden Kunden im System.
- „NQ_scv()“ oder „Queue_scv()“ oder „Schlange_scv()“ oder „Warteschlange_scv()“:
Liefert den quadrierten Variationskoeffizienten der Anzahl an wartenden Kunden im System.

12.3 Kunden an den Stationen

12.3.1 Anzahl an Kunden an einer Station

- „WIP(id)“ oder „N(id)“ oder „Station(id)“:
Liefert die aktuelle Gesamtanzahl an Kunden an Station *id*.
- „WIP(id1;id2)“ oder „N(id1;id2)“ oder „Station(id1;id2)“:
Liefert die aktuelle Gesamtanzahl an Kunden an Station *id1*. Wobei nur Kunden von dem Typ, dessen Name an Quelle bzw. Namenszuweisung *id2* auftritt, berücksichtigt werden.
- „WIP_avg(id)“ oder „Station_avg(id)“ oder „N_avg(id)“ oder „WIP_Mittelwert(id)“ oder „Station_Mittelwert(id)“ oder „N_Mittelwert(id)“:
Liefert die mittlere Anzahl an Kunden an Station *id*.
- „WIP_median(id)“ oder „Station_median(id)“ oder „N_median(id)“:
Liefert den Median der Anzahl an Kunden an Station *id*.
- „WIP_quantil(p;id)“ oder „Station_quantil(p;id)“ oder „N_quantil(p;id)“:
Liefert das Quantil zur Wahrscheinlichkeit *p* der Anzahl an Kunden an Station *id*.
- „WIP_min(id)“ oder „Station_min(id)“ oder „N_min(id)“ oder „WIP_Minimum(id)“ oder „Station_Minimum(id)“ oder „N_Minimum(id)“:
Liefert die minimale Anzahl an Kunden an Station *id*.
- „WIP_max(id)“ oder „Station_max(id)“ oder „N_max(id)“ oder „WIP_Maximum(id)“ oder „Station_Maximum(id)“ oder „N_Maximum(id)“:
Liefert die maximale Anzahl an Kunden an Station *id*.
- „WIP_var(id)“ oder „Station_var(id)“ oder „N_var(id)“ oder „WIP_Varianz(id)“ oder „Station_Varianz(id)“ oder „N_Varianz(id)“:
Liefert die Varianz der Anzahl an Kunden an Station *id*.
- „WIP_sd(id)“ oder „Station_sd(id)“ oder „N_sd(id)“ oder „WIP_Standardabweichung(id)“ oder „Station_Standardabweichung(id)“ oder „N_Standardabweichung(id)“:
Liefert die Standardabweichung der Anzahl an Kunden an Station *id*.
- „WIP_cv(id)“ oder „Station_cv(id)“ oder „N_cv(id)“:
Liefert den Variationskoeffizienten der Anzahl an Kunden an Station *id*.
- „WIP_scv(id)“ oder „Station_scv(id)“ oder „N_scv(id)“:
Liefert den quadrierten Variationskoeffizienten der Anzahl an Kunden an Station *id*.
- „WIP_hist(id;state)“ oder „Station_hist(id;state)“ oder „N_hist(id;state)“:
Liefert den Anteil der Zeit, in der sich das System in Bezug auf die Anzahl an Kunden an Station *id* im Zustand *state* befunden hat.
- „WIP_hist(id;stateA;stateB)“ oder „Station_hist(id;stateA;stateB)“ oder „N_hist(id;stateA;stateB)“:
Liefert den Anteil der Zeit, in der sich das System in Bezug auf die Anzahl an Kunden an Station *id* im Zustand *stateA* oder *stateB* befunden hat.

Liefert den Anteil der Zeit, in der sich das System in Bezug auf die Anzahl an Kunden an Station **id** in einem Zustand größer als **stateA** und kleiner oder gleich **stateB** befunden hat.

12.3.2 Anzahl an Kunden in der Warteschlange an einer Station

- „NQ(id)” oder „Queue(id)” oder „Schlange(id)” oder „Warteschlange(id)”:
Liefert die aktuelle Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ(id;nr)” oder „Queue(id;nr)” oder „Schlange(id;nr)” oder „Warteschlange(id;nr)”:
Liefert die aktuelle Anzahl an Kunden in der Teilwarteschlange **nr** (1-basierend) an Station **id**. (Dieser Befehl kann nur auf „Zusammenführen“-Elemente angewandt werden.)
- „NQ_avg(id)” oder „Queue_avg(id)” oder „Schlange_avg(id)” oder „Warteschlange_avg(id)” oder „NQ_Mittelwert(id)” oder „Queue_Mittelwert(id)” oder „Schlange_Mittelwert(id)” oder „Warteschlange_Mittelwert(id)”:
Liefert die mittlere Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ_median(id)” oder „Queue_median(id)” oder „Schlange_median(id)” oder „Warteschlange_median(id)”:
Liefert den Median der Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ_quantil(p;id)” oder „Queue_quantil(p;id)” oder „Schlange_quantil(p;id)” oder „Warteschlange_quantil(p;id)”:
Liefert das Quantil zur Wahrscheinlichkeit **p** der Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ_min(id)” oder „Queue_min(id)” oder „Schlange_min(id)” oder „Warteschlange_min(id)” oder „NQ_Minimum(id)” oder „Queue_Minimum(id)” oder „Schlange_Minimum(id)” oder „Warteschlange_Minimum(id)”:
Liefert die minimale Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ_max(id)” oder „Queue_max(id)” oder „Schlange_max(id)” oder „Warteschlange_max(id)” oder „NQ_Maximum(id)” oder „Queue_Maximum(id)” oder „Schlange_Maximum(id)” oder „Warteschlange_Maximum(id)”:
Liefert die maximale Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ_var(id)” oder „Queue_var(id)” oder „Schlange_var(id)” oder „Warteschlange_var(id)” oder „NQ_Varianz(id)” oder „Queue_Varianz(id)” oder „Schlange_Varianz(id)” oder „Warteschlange_Varianz(id)”:
Liefert die Varianz der Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ_sd(id)” oder „Queue_sd(id)” oder „Schlange_sd(id)” oder „Warteschlange_sd(id)” oder „NQ_Standardabweichung(id)” oder „Queue_Standardabweichung(id)” oder „Schlange_Standardabweichung(id)” oder „Warteschlange_Standardabweichung(id)”:
Liefert die Standardabweichung der Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ_cv(id)” oder „Queue_cv(id)” oder „Schlange_cv(id)” oder „Warteschlange_cv(id)”:
Liefert den Variationskoeffizienten der Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ_scv(id)” oder „Queue_scv(id)” oder „Schlange_scv(id)” oder „Warteschlange_scv(id)”:
Liefert den quadrierten Variationskoeffizienten der Anzahl an Kunden in der Warteschlange an Station **id**.
- „NQ_hist(id;state)” oder „Queue_hist(id;state)” oder „Schlange_hist(id;state)” oder „Warteschlange_hist(id;state)”:
Liefert den Anteil der Zeit, in der sich **state** Kunden in der Warteschlange an Station **id** befunden haben.

- „NQ_hist(id;stateA;stateB)” oder „Queue_hist(id;stateA;stateB)” oder „Schlange_hist(id;stateA;stateB)” oder „Warteschlange_hist(id;stateA;stateB)”:
Liefert den Anteil der Zeit, in der sich mehr als **stateA** und höchstens **stateB** Kunden in der Warteschlange an Station **id** befunden haben.

12.3.3 Anzahl an Kunden in Bedienung an einer Station

- „Process(id)”:
Liefert die aktuelle Anzahl an Kunden, die gerade an Station **id** bedient werden.

12.3.4 Anzahl an Ankünften und Abhängen an einer Station

- „NumberIn(id)” oder „CountIn(id)”:
Liefert die Anzahl an Kundenankünften an Station **id**.
- „NumberOut(id)” oder „CountOut(id)”:
Liefert die Anzahl an Kundenabgängen von Station **id**.

12.4 Kunden nach Kundentypen

12.4.1 Anzahl an Kunden im System nach Kundentypen

- „WIP(id)” oder „N(id)” oder „Station(id)”:
Liefert die aktuelle Gesamtanzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „WIP_avg(id)” oder „Station_avg(id)” oder „N_avg(id)” oder „WIP_Mittelwert(id)” oder „Station_Mittelwert(id)” oder „N_Mittelwert(id)”:
Liefert die mittlere Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „WIP_median(id)” oder „Station_median(id)” oder „N_median(id)”:
Liefert den Median der Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „WIP_quantil(p;id)” oder „Station_quantil(p;id)” oder „N_quantil(p;id)”:
Liefert das Quantil zur Wahrscheinlichkeit **p** der Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „WIP_min(id)” oder „Station_min(id)” oder „N_min(id)” oder „WIP_Minimum(id)” oder „Station_Minimum(id)” oder „N_Minimum(id)”:
Liefert die minimale Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „WIP_max(id)” oder „Station_max(id)” oder „N_max(id)” oder „WIP_Maximum(id)” oder „Station_Maximum(id)” oder „N_Maximum(id)”:
Liefert die maximale Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „WIP_var(id)” oder „Station_var(id)” oder „N_var(id)” oder „WIP_Varianz(id)” oder „Station_Varianz(id)” oder „N_Varianz(id)”:
Liefert die Varianz der Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „WIP_sd(id)” oder „Station_sd(id)” oder „N_sd(id)” oder „WIP_Standardabweichung(id)” oder „Station_Standardabweichung(id)” oder „N_Standardabweichung(id)”:
Liefert die Standardabweichung der Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.

- „WIP_cv(id)” oder „Station_cv(id)” oder „N_cv(id)”:
Liefert den Variationskoeffizienten der Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „WIP_scv(id)” oder „Station_scv(id)” oder „N_scv(id)”:
Liefert den quadrierten Variationskoeffizienten der Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „WIP_hist(id;state)” oder „Station_hist(id;state)” oder „N_hist(id;state)”:
Liefert den Anteil der Zeit, in der sich das System in Bezug auf die Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt, im Zustand **state** befunden hat.
- „WIP_hist(id;stateA;stateB)” oder „Station_hist(id;stateA;stateB)” oder „N_hist(id;stateA;stateB)”:
Liefert den Anteil der Zeit, in der sich das System in Bezug auf die Anzahl an Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt, in einem Zustand größer als **stateA** und kleiner oder gleich **stateB** befunden hat.

12.4.2 Anzahl an wartenden Kunden im System nach Kundentypen

- „NQ(id)” oder „Queue(id)” oder „Schlange(id)” oder „Warteschlange(id)”:
Liefert die aktuelle Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „NQ_avg(id)” oder „Queue_avg(id)” oder „Schlange_avg(id)” oder „Warteschlange_avg(id)” oder „NQ_Mittelwert(id)” oder „Queue_Mittelwert(id)” oder „Schlange_Mittelwert(id)” oder „Warteschlange_Mittelwert(id)”:
Liefert die mittlere Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „NQ_median(id)” oder „Queue_median(id)” oder „Schlange_median(id)” oder „Warteschlange_median(id)”:
Liefert den Median der Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „NQ_quantil(p;id)” oder „Queue_quantil(p;id)” oder „Schlange_quantil(p;id)” oder „Warteschlange_quantil(p;id)”:
Liefert das Quantil zur Wahrscheinlichkeit **p** der Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „NQ_min(id)” oder „Queue_min(id)” oder „Schlange_min(id)” oder „Warteschlange_min(id)” oder „NQ_Minimum(id)” oder „Queue_Minimum(id)” oder „Schlange_Minimum(id)” oder „Warteschlange_Minimum(id)”:
Liefert die minimale Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „NQ_max(id)” oder „Queue_max(id)” oder „Schlange_max(id)” oder „Warteschlange_max(id)” oder „NQ_Maximum(id)” oder „Queue_Maximum(id)” oder „Schlange_Maximum(id)” oder „Warteschlange_Maximum(id)”:
Liefert die maximale Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „NQ_var(id)” oder „Queue_var(id)” oder „Schlange_var(id)” oder „Warteschlange_var(id)” oder „NQ_Varianz(id)” oder „Queue_Varianz(id)” oder „Schlange_Varianz(id)” oder „Warteschlange_Varianz(id)”:
Liefert die Varianz der Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.

Liefert die Varianz der Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.

- „NQ_sd(id)” oder „Queue_sd(id)” oder „Schlange_sd(id)” oder „Warteschlange_sd(id)” oder „NQ_Standardabweichung(id)” oder „Queue_Standardabweichung(id)” oder „Schlange_Standardabweichung(id)” oder „Warteschlange_Standardabweichung(id)”: Liefert die Standardabweichung der Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „NQ_cv(id)” oder „Queue_cv(id)” oder „Schlange_cv(id)” oder „Warteschlange_cv(id)”: Liefert den Variationskoeffizienten der Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „NQ_scv(id)” oder „Queue_scv(id)” oder „Schlange_scv(id)” oder „Warteschlange_scv(id)”: Liefert den quadrierten Variationskoeffizienten der Anzahl an wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- „NQ_hist(id;state)” oder „Queue_hist(id;state)” oder „Schlange_hist(id;state)” oder „Warteschlange_hist(id;state)”: Liefert den Anteil der Zeit, in der sich **state** wartende Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt, im System befunden haben.
- „NQ_hist(id;stateA;stateB)” oder „Queue_hist(id;stateA;stateB)” oder „Schlange_hist(id;stateA;stateB)” oder „Warteschlange_hist(id;stateA;stateB)”: Liefert den Anteil der Zeit, in der sich mehr als **stateA** und höchsten **stateB** wartenden Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt, im System befunden haben.

12.5 Zähler und Durchsatz

- „Zähler(id)” oder „Counter(id)” oder „Value(id)” oder „Wert(id)”: Liefert den Wert des Zählers in Station **id**. (Kann nur auf „Differenzzähler“- auf „Zähler“- und „Durchsatz“-Elemente angewandt werden.)
- „Anteil(id)” oder „Part(id)”: Liefert den Anteil des Zählerwertes innerhalb der Zählergruppe, in der er sich befindet. (Kann nur auf „Zähler“-Elemente angewandt werden.)

12.6 Wartezeiten

12.6.1 Wartezeiten an einer Station

- „Wartezeit_sum(id)” oder „Wartezeit_gesamt(id)” oder „Wartezeit_summe(id)”: Liefert die Summe der an Station **id** bisher angefallenen Wartezeiten (in Sekunden).
- „Wartezeit_avg(id)” oder „Wartezeit_average(id)” oder „Wartezeit_Mittelwert(id)”: Liefert die mittlere Wartezeit der Kunden an Station **id** (in Sekunden).
- „Wartezeit_median(id)”: Liefert den Median der Wartezeiten der Kunden an Station **id** (in Sekunden).
- „Wartezeit_quantil(p;id)”: Liefert das Quantil zur Wahrscheinlichkeit **p** der Wartezeiten der Kunden an Station **id** (in Sekunden).
- „Wartezeit_min(id)” oder „Wartezeit_Minimum(id)”: Liefert die minimale Wartezeit der Kunden an Station **id** (in Sekunden).

- **„Wartezeit_max(id)”** oder **„Wartezeit_Maximum(id)”**:
Liefert die maximale Wartezeit der Kunden an Station **id** (in Sekunden).
- **„Wartezeit_var(id)”** oder **„Wartezeit_Varianz(id)”**:
Liefert die Varianz der Wartezeiten der Kunden an Station **id** (bezogen auf Sekunden).
- **„Wartezeit_sd(id)”** oder **„Wartezeit_Standardabweichung(id)”**:
Liefert die Standardabweichung der Wartezeiten der Kunden an Station **id** (bezogen auf Sekunden).
- **„Wartezeit_cv(id)”**:
Liefert den Variationskoeffizienten der Wartezeiten der Kunden an Station **id**.
- **„Wartezeit_scv(id)”**:
Liefert den quadrierten Variationskoeffizienten der Wartezeiten der Kunden an Station **id**.
- **„Wartezeit_hist(id;time)”**:
Liefert den Anteil der Kunden, für den die Wartezeit an Station **id** **time** Sekunden gedauert hat.
- **„Wartezeit_hist(id;timeA;timeB)”**:
Liefert den Anteil der Kunden, für den die Wartezeit an Station **id** mehr als **timeA** und höchstens **timeB** Sekunden gedauert hat.

12.6.2 Wartezeiten über alle Kundentypen

- **„Wartezeit_avg()”** oder **„Wartezeit_average()”** oder **„Wartezeit_Mittelwert()”**:
Liefert die mittlere Wartezeit über alle Kunden (in Sekunden).
- **„Wartezeit_median()”**:
Liefert den Median der Wartezeiten über alle Kunden (in Sekunden).
- **„Wartezeit_quntil(p)”**:
Liefert das Quantil zur Wahrscheinlichkeit **p** der Wartezeiten über alle Kunden (in Sekunden).
- **„Wartezeit_min()”** oder **„Wartezeit_Minimum()”**:
Liefert die minimale Wartezeit über alle Kunden (in Sekunden).
- **„Wartezeit_max()”** oder **„Wartezeit_Maximum()”**:
Liefert die maximale Wartezeit über alle Kunden (in Sekunden).
- **„Wartezeit_var()”** oder **„Wartezeit_Varianz()”**:
Liefert die Varianz der Wartezeiten über alle Kunden (bezogen auf Sekunden).
- **„Wartezeit_sd()”** oder **„Wartezeit_Standardabweichung()”**:
Liefert die Standardabweichung der Wartezeiten über alle Kunden (bezogen auf Sekunden).
- **„Wartezeit_cv()”**:
Liefert den Variationskoeffizienten der Wartezeiten über alle Kunden.
- **„Wartezeit_scv()”**:
Liefert den quadrierten Variationskoeffizienten über alle Kunden.
- **„Wartezeit_histAll(time)”**:
Liefert den Anteil der Kunden, für den die Wartezeit **time** Sekunden gedauert hat.
- **„Wartezeit_histAll(timeA;timeB)”**:
Liefert den Anteil der Kunden, für den die Wartezeit mehr als **timeA** und höchstens **timeB** Sekunden gedauert hat.

12.6.3 Wartezeiten für einen Kundentypen

- **„Wartezeit_sum(id)“** oder **„Wartezeit_gesamt(id)“** oder **„Wartezeit_summe(id)“**: Liefert die Summe der Wartezeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Wartezeit_avg(id)“** oder **„Wartezeit_average(id)“** oder **„Wartezeit_Mittelwert(id)“**: Liefert die mittlere Wartezeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Wartezeit_median(id)“**: Liefert den Median der Wartezeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Wartezeit_quantil(p;id)“**: Liefert das Quantil zur Wahrscheinlichkeit **p** der Wartezeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Wartezeit_min(id)“** oder **„Wartezeit_Minimum(id)“**: Liefert die minimale Wartezeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Wartezeit_max(id)“** oder **„Wartezeit_Maximum(id)“**: Liefert die maximale Wartezeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Wartezeit_var(id)“** oder **„Wartezeit_Varianz(id)“**: Liefert die Varianz der Wartezeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (bezogen auf Sekunden).
- **„Wartezeit_sd(id)“** oder **„Wartezeit_Standardabweichung(id)“**: Liefert die Standardabweichung der Wartezeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (bezogen auf Sekunden).
- **„Wartezeit_cv(id)“**: Liefert den Variationskoeffizienten der Wartezeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- **„Wartezeit_scv(id)“**: Liefert den quadrierten Variationskoeffizienten der Wartezeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.

12.7 Transferzeiten

12.7.1 Transferzeiten an einer Station

- **„Transferzeit_sum(id)“** oder **„Transferzeit_gesamt(id)“** oder **„Transferzeit_summe(id)“**: Liefert die Summe der an Station **id** bisher angefallenen Transferzeiten (in Sekunden).
- **„Transferzeit_avg(id)“** oder **„Transferzeit_average(id)“** oder **„Transferzeit_Mittelwert(id)“**: Liefert die mittlere Transferzeit der Kunden an Station **id** (in Sekunden).
- **„Transferzeit_median(id)“**: Liefert den Median der Transferzeiten der Kunden an Station **id** (in Sekunden).

- **„Transferzeit_quantil(p;id)“**:
Liefert das Quantil zur Wahrscheinlichkeit *p* der Transferzeiten der Kunden an Station *id* (in Sekunden).
- **„Transferzeit_min(id)“** oder **„Transferzeit_Minimum(id)“**:
Liefert die minimale Transferzeit der Kunden an Station *id* (in Sekunden).
- **„Transferzeit_max(id)“** oder **„Transferzeit_Maximum(id)“**:
Liefert die maximale Transferzeit der Kunden an Station *id* (in Sekunden).
- **„Transferzeit_var(id)“** oder **„Transferzeit_Varianz(id)“**:
Liefert die Varianz der Transferzeiten der Kunden an Station *id* (bezogen auf Sekunden).
- **„Transferzeit_sd(id)“** oder **„Transferzeit_Standardabweichung(id)“**:
Liefert die Standardabweichung der Transferzeiten der Kunden an Station *id* (bezogen auf Sekunden).
- **„Transferzeit_cv(id)“**:
Liefert den Variationskoeffizienten der Transferzeiten der Kunden an Station *id*.
- **„Transferzeit_scv(id)“**:
Liefert den quadrierten Variationskoeffizienten der Transferzeiten der Kunden an Station *id*.
- **„Transferzeit_hist(id;time)“**:
Liefert den Anteil der Kunden, für den die Transferzeit an Station *id* *time* Sekunden gedauert hat.
- **„Transferzeit_hist(id;timeA;timeB)“**:
Liefert den Anteil der Kunden, für den die Transferzeit an Station *id* mehr als *timeA* und höchstens *timeB* Sekunden gedauert hat.

12.7.2 Transferzeiten über alle Kundentypen

- **„Transferzeit_avg()“** oder **„Transferzeit_average()“** oder **„Transferzeit_Mittelwert()“**:
Liefert die mittlere Transferzeit über alle Kunden (in Sekunden).
- **„Transferzeit_median()“**:
Liefert den Median der Transferzeiten über alle Kunden (in Sekunden).
- **„Transferzeit_quantil(p)“**:
Liefert das Quantil zur Wahrscheinlichkeit *p* der Transferzeiten über alle Kunden (in Sekunden).
- **„Transferzeit_min()“** oder **„Transferzeit_Minimum()“**:
Liefert die minimale Transferzeit über alle Kunden (in Sekunden).
- **„Transferzeit_max()“** oder **„Transferzeit_Maximum()“**:
Liefert die maximale Transferzeit über alle Kunden (in Sekunden).
- **„Transferzeit_var()“** oder **„Transferzeit_Varianz()“**:
Liefert die Varianz der Transferzeiten über alle Kunden (bezogen auf Sekunden).
- **„Transferzeit_sd()“** oder **„Transferzeit_Standardabweichung()“**:
Liefert die Standardabweichung der Transferzeiten über alle Kunden (bezogen auf Sekunden).
- **„Transferzeit_cv()“**:
Liefert den Variationskoeffizienten der Transferzeiten über alle Kunden.
- **„Transferzeit_scv()“**:
Liefert den quadrierten Variationskoeffizienten der Transferzeiten über alle Kunden.
- **„Transferzeit_histAll(time)“**:
Liefert den Anteil der Kunden, für den die Transferzeit *time* Sekunden gedauert hat.

- **„Transferzeit_histAll(timeA;timeB)“:**
Liefert den Anteil der Kunden, für den die Transferzeit mehr als **timeA** und höchstens **timeB** Sekunden gedauert hat.

12.7.3 Transferzeiten für einen Kundentypen

- **„Transferzeit_sum(id)“ oder „Transferzeit_gesamt(id)“ oder „Transferzeit_summe(id)“:**
Liefert die Summe der Transferzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Transferzeit_avg(id)“ oder „Transferzeit_average(id)“ oder „Transferzeit_Mittelwert(id)“:**
Liefert die mittlere Transferzeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Transferzeit_median(id)“:**
Liefert den Median der Transferzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Transferzeit_quantil(p;id)“:**
Liefert das Quantil zur Wahrscheinlichkeit **p** der Transferzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Transferzeit_min(id)“ oder „Transferzeit_Minimum(id)“:**
Liefert die minimale Transferzeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Transferzeit_max(id)“ oder „Transferzeit_Maximum(id)“:**
Liefert die maximale Transferzeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Transferzeit_var(id)“ oder „Transferzeit_Varianz(id)“:**
Liefert die Varianz der Transferzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (bezogen auf Sekunden).
- **„Transferzeit_sd(id)“ oder „Transferzeit_Standardabweichung(id)“:**
Liefert die Standardabweichung der Transferzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (bezogen auf Sekunden).
- **„Transferzeit_cv(id)“:**
Liefert den Variationskoeffizienten der Transferzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- **„Transferzeit_scv(id)“:**
Liefert den quadrierten Variationskoeffizienten der Transferzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.

12.8 Bedienzeiten

12.8.1 Bedienzeiten an einer Station

- **„Bedienzeit_sum(id)“ oder „Bedienzeit_gesamt(id)“ oder „Bedienzeit_summe(id)“:**
Liefert die Summe der an Station **id** bisher angefallenen Bedienzeiten (in Sekunden).
- **„Bedienzeit_avg(id)“ oder „Bedienzeit_average(id)“ oder „Bedienzeit_Mittelwert(id)“:**
Liefert die mittlere Bedienzeit der Kunden an Station **id** (in Sekunden).

- **„Bedienzeit_median(id)”**:
Liefert den Median der Bedienzeiten der Kunden an Station **id** (in Sekunden).
- **„Bedienzeit_quantil(p;id)”**:
Liefert das Quantil zur Wahrscheinlichkeit **p** der Bedienzeiten der Kunden an Station **id** (in Sekunden).
- **„Bedienzeit_min(id)”** oder **„Bedienzeit_Minimum(id)”**:
Liefert die minimale Bedienzeit der Kunden an Station **id** (in Sekunden).
- **„Bedienzeit_max(id)”** oder **„Bedienzeit_Maximum(id)”**:
Liefert die maximale Bedienzeit der Kunden an Station **id** (in Sekunden).
- **„Bedienzeit_var(id)”** oder **„Bedienzeit_Varianz(id)”**:
Liefert die Varianz der Bedienzeiten der Kunden an Station **id** (bezogen auf Sekunden).
- **„Bedienzeit_sd(id)”** oder **„Bedienzeit_Standardabweichung(id)”**:
Liefert die Standardabweichung der Bedienzeiten der Kunden an Station **id** (bezogen auf Sekunden).
- **„Bedienzeit_cv(id)”**:
Liefert den Variationskoeffizienten der Bedienzeiten der Kunden an Station **id**.
- **„Bedienzeit_scv(id)”**:
Liefert den quadrierten Variationskoeffizienten der Bedienzeiten der Kunden an Station **id**.
- **„Bedienzeit_hist(id;time)”**:
Liefert den Anteil der Kunden, für den die Bedienzeit an Station **id** **time** Sekunden gedauert hat.
- **„Bedienzeit_hist(id;timeA;timeB)”**:
Liefert den Anteil der Kunden, für den die Bedienzeit an Station **id** mehr als **timeA** und höchstens **timeB** Sekunden gedauert hat.

12.8.2 Bedienzeiten über alle Kundentypen

- **„Bedienzeit_avg()”** oder **„Bedienzeit_average()”** oder **„Bedienzeit_Mittelwert()”**:
Liefert die mittlere Bedienzeit über alle Kunden (in Sekunden).
- **„Bedienzeit_median()”**:
Liefert den Median der Bedienzeiten über alle Kunden (in Sekunden).
- **„Bedienzeit_quantil(p)”**:
Liefert das Quantil zur Wahrscheinlichkeit **p** der Bedienzeiten über alle Kunden (in Sekunden).
- **„Bedienzeit_min()”** oder **„Bedienzeit_Minimum()”**:
Liefert die mittlere Bedienzeit über alle Kunden (in Sekunden).
- **„Bedienzeit_max()”** oder **„Bedienzeit_Maximum()”**:
Liefert die maximale Bedienzeit über alle Kunden (in Sekunden).
- **„Bedienzeit_var()”** oder **„Bedienzeit_Varianz()”**:
Liefert die Varianz der Bedienzeiten über alle Kunden (bezogen auf Sekunden).
- **„Bedienzeit_sd()”** oder **„Bedienzeit_Standardabweichung()”**:
Liefert die Standardabweichung der Bedienzeiten über alle Kunden (bezogen auf Sekunden).
- **„Bedienzeit_cv()”**:
Liefert den Variationskoeffizienten der Bedienzeiten über alle Kunden.
- **„Bedienzeit_scv()”**:
Liefert den quadrierten Variationskoeffizienten der Bedienzeiten über alle Kunden.
- **„Bedienzeit_histAll(time)”**:
Liefert den Anteil der Kunden, für den die Bedienzeit **time** Sekunden gedauert hat.

- **„Bedienzeit_histAll(timeA;timeB)“:**
Liefert den Anteil der Kunden, für den die Bedienzeit mehr als **timeA** und höchstens **timeB** Sekunden gedauert hat.

12.8.3 Bedienzeiten für einen Kundentypen

- **„Bedienzeit_sum(id)“** oder **„Bedienzeit_gesamt(id)“** oder **„Bedienzeit_summe(id)“:**
Liefert die Summe der Bedienzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Bedienzeit_avg(id)“** oder **„Bedienzeit_average(id)“** oder **„Bedienzeit_Mittelwert(id)“:**
Liefert die mittlere Bedienzeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Bedienzeit_median(id)“:**
Liefert den Median der Bedienzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Bedienzeit_quantil(p;id)“:**
Liefert das Quantil zur Wahrscheinlichkeit **p** der Bedienzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Bedienzeit_min(id)“** oder **„Bedienzeit_Minimum(id)“:**
Liefert die minimale Bedienzeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Bedienzeit_max(id)“** oder **„Bedienzeit_Maximum(id)“:**
Liefert die maximale Bedienzeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Bedienzeit_var(id)“** oder **„Bedienzeit_Varianz(id)“:**
Liefert die Varianz der Bedienzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (bezogen auf Sekunden).
- **„Bedienzeit_sd(id)“** oder **„Bedienzeit_Standardabweichung(id)“:**
Liefert die Standardabweichung der Bedienzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (bezogen auf Sekunden).
- **„Bedienzeit_cv(id)“:**
Liefert den Variationskoeffizienten der Bedienzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- **„Bedienzeit_scv(id)“:**
Liefert den quadrierten Variationskoeffizienten der Bedienzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.

12.9 Verweilzeiten

12.9.1 Verweilzeiten an einer Station

- **„Verweilzeit_sum(id)“** oder **„Verweilzeit_gesamt(id)“** oder **„Verweilzeit_summe(id)“:**
Liefert die Summe der an Station **id** bisher angefallenen Verweilzeiten (in Sekunden).
- **„Verweilzeit_avg(id)“** oder **„Verweilzeit_average(id)“** oder **„Verweilzeit_Mittelwert(id)“:**
Liefert die mittlere Verweilzeit der Kunden an Station **id** (in Sekunden).

- **„Verweilzeit_median(id)”**:
Liefert den Median der Verweilzeiten der Kunden an Station **id** (in Sekunden).
- **„Verweilzeit_quantil(p;id)”**:
Liefert das Quantil zur Wahrscheinlichkeit **p** der Verweilzeiten der Kunden an Station **id** (in Sekunden).
- **„Verweilzeit_min(id)”** oder **„Verweilzeit_Minimum(id)”**:
Liefert die minimale Verweilzeit der Kunden an Station **id** (in Sekunden).
- **„Verweilzeit_max(id)”** oder **„Verweilzeit_Maximum(id)”**:
Liefert die maximale Verweilzeit der Kunden an Station **id** (in Sekunden).
- **„Verweilzeit_var(id)”** oder **„Verweilzeit_Varianz(id)”**:
Liefert die Varianz der Verweilzeiten der Kunden an Station **id** (bezogen auf Sekunden).
- **„Verweilzeit_sd(id)”** oder **„Verweilzeit_Standardabweichung(id)”**:
Liefert die Standardabweichung der Verweilzeiten der Kunden an Station **id** (bezogen auf Sekunden).
- **„Verweilzeit_cv(id)”**:
Liefert den Variationskoeffizienten der Verweilzeiten der Kunden an Station **id**.
- **„Verweilzeit_scv(id)”**:
Liefert den quadrierten Variationskoeffizienten der Verweilzeiten der Kunden an Station **id**.
- **„Verweilzeit_hist(id;time)”**:
Liefert den Anteil der Kunden, für den die Verweilzeit an Station **id** **time** Sekunden gedauert hat.
- **„Verweilzeit_hist(id;timeA;timeB)”**:
Liefert den Anteil der Kunden, für den die Verweilzeit an Station **id** mehr als **timeA** und höchstens **timeB** Sekunden gedauert hat.

12.9.2 Verweilzeiten über alle Kundentypen

- **„Verweilzeit_avg()”** oder **„Verweilzeit_average()”** oder **„Verweilzeit_Mittelwert()”**:
Liefert die mittlere Verweilzeit über alle Kunden (in Sekunden).
- **„Verweilzeit_median()”**:
Liefert den Median der Verweilzeiten über alle Kunden (in Sekunden).
- **„Verweilzeit_quantil(p)”**:
Liefert das Quantil zur Wahrscheinlichkeit **p** der Verweilzeiten über alle Kunden (in Sekunden).
- **„Verweilzeit_min()”** oder **„Verweilzeit_Minimum()”**:
Liefert die minimale Verweilzeit über alle Kunden (in Sekunden).
- **„Verweilzeit_max()”** oder **„Verweilzeit_Maximum()”**:
Liefert die maximale Verweilzeit über alle Kunden (in Sekunden).
- **„Verweilzeit_var()”** oder **„Verweilzeit_Varianz()”**:
Liefert die Varianz der Verweilzeiten über alle Kunden (bezogen auf Sekunden).
- **„Verweilzeit_sd()”** oder **„Verweilzeit_Standardabweichung()”**:
Liefert die Standardabweichung der Verweilzeiten über alle Kunden (bezogen auf Sekunden).
- **„Verweilzeit_cv()”**:
Liefert den Variationskoeffizienten der Verweilzeiten über alle Kunden.
- **„Verweilzeit_scv()”**:
Liefert den quadrierten Variationskoeffizienten über alle Kunden.
- **„Verweilzeit_histAll(time)”**:
Liefert den Anteil der Kunden, für den die Verweilzeit **time** Sekunden gedauert hat.

- **„Verweilzeit_histAll(timeA;timeB)“**:
Liefert den Anteil der Kunden, für den die Verweilzeit mehr als **timeA** und höchstens **timeB** Sekunden gedauert hat.

12.9.3 Verweilzeiten für einen Kundentypen

- **„Verweilzeit_sum(id)“** oder **„Verweilzeit_gesamt(id)“** oder **„Verweilzeit_summe(id)“**: Liefert die Summe der Verweilzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Verweilzeit_avg(id)“** oder **„Verweilzeit_average(id)“** oder **„Verweilzeit_Mittelwert(id)“**: Liefert die mittlere Verweilzeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Verweilzeit_median(id)“**:
Liefert den Median der Verweilzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Verweilzeit_quantil(p;id)“**:
Liefert das Quantil zur Wahrscheinlichkeit **p** der Verweilzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Verweilzeit_min(id)“** oder **„Verweilzeit_Minimum(id)“**:
Liefert die minimale Verweilzeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Verweilzeit_max(id)“** oder **„Verweilzeit_Maximum(id)“**:
Liefert die maximale Verweilzeit der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (in Sekunden).
- **„Verweilzeit_var(id)“** oder **„Verweilzeit_Varianz(id)“**:
Liefert die Varianz der Verweilzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (bezogen auf Sekunden).
- **„Verweilzeit_sd(id)“** oder **„Verweilzeit_Standardabweichung(id)“**:
Liefert die Standardabweichung der Verweilzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt (bezogen auf Sekunden).
- **„Verweilzeit_cv(id)“**:
Liefert den Variationskoeffizienten der Verweilzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.
- **„Verweilzeit_scv(id)“**:
Liefert den quadrierten Variationskoeffizienten der Verweilzeiten der Kunden, deren Name an Quelle bzw. Namenszuweisung **id** auftritt.

12.10 Auslastung der Ressourcen

12.10.1 Auslastung einer Ressource

- **„resource_count(id)“** oder **„resource_capacity(id)“** oder **„MR(id)“**:
Liefert die Anzahl der momentan vorhandenen Bediener in der angegebenen Ressource.
- **„resource_down(id)“**:
Liefert die Anzahl der momentan in Ausfallzeit befindlichen Bediener in der angegebenen Ressource.

- `„resource(id)”` oder `„utilization(id)”` oder `„NR(id)”`:
Liefert die Anzahl der momentan belegten Bediener in der angegebenen Ressource.
- `„resource_avg(id)”` oder `„resource_average(id)”` oder `„resource_Mittelwert(id)”` oder `„utilization_avg(id)”` oder `„utilization_average(id)”` oder `„utilization_Mittelwert(id)”`:
Liefert die mittlere Anzahl an belegten Bedienern in der angegebenen Ressource.
- `„resource_min(id)”` oder `„resource_Minimum(id)”` oder `„utilization_min(id)”` oder `„utilization_Minimum(id)”`:
Liefert die minimale Anzahl an belegten Bedienern in der angegebenen Ressource.
- `„resource_max(id)”` oder `„resource_Maximum(id)”` oder `„utilization_max(id)”` oder `„utilization_Maximum(id)”`:
Liefert die maximale Anzahl an belegten Bedienern in der angegebenen Ressource.
- `„resource_var(id)”` oder `„resource_Varianz(id)”` oder `„utilization_var(id)”` oder `„utilization_Varianz(id)”`:
Liefert die Varianz der Anzahl an belegten Bedienern in der angegebenen Ressource.
- `„resource_sd(id)”` oder `„resource_Standardabweichung(id)”` oder `„utilization_sd(id)”` oder `„utilization_Standardabweichung(id)”`:
Liefert die Standardabweichung der Anzahl an belegten Bedienern in der angegebenen Ressource.
- `„resource_cv(id)”` oder `„utilization_cv(id)”`:
Liefert den Variationskoeffizienten der Anzahl an belegten Bedienern in der angegebenen Ressource.
- `„resource_scv(id)”` oder `„utilization_scv(id)”`:
Liefert den quadrierten Variationskoeffizienten der Anzahl an belegten Bedienern in der angegebenen Ressource.
- `„resource_hist(id;state)”` oder `„utilization_hist(id;state)”`:
Liefert den Anteil der Zeit, in der **state** Bediener der angegebenen Ressource ausgelastet waren.
- `„resource_hist(id;stateA;stateB)”` oder `„utilization_hist(id;stateA;stateB)”`:
Liefert den Anteil der Zeit, mehr als **stateA** und höchstens **stateB** Bediener der angegebenen Ressource ausgelastet waren.

12.10.2 Auslastung aller Ressourcen zusammen

- `„resource_count()”` oder `„resource_capacity()”` oder `„MR()”`:
Liefert die Anzahl der momentan vorhandenen Bedienern in allen Ressourcen zusammen.
- `„resource_down()”`:
Liefert die Anzahl der momentan in Ausfallzeit befindlichen Bediener in allen Ressourcen zusammen.
- `„resource()”` oder `„utilization()”` oder `„NR()”`:
Liefert die Anzahl der momentan belegten Bediener in allen Ressourcen zusammen.
- `„resource_avg()”` oder `„resource_average()”` oder `„resource_Mittelwert()”` oder `„utilization_avg()”` oder `„utilization_average()”` oder `„utilization_Mittelwert()”`:
Liefert die mittlere Anzahl an belegten Bedienern in allen Ressourcen zusammen.
- `„resource_min()”` oder `„resource_Minimum()”` oder `„utilization_min()”` oder `„utilization_Minimum()”`:
Liefert die minimale Anzahl an belegten Bedienern in allen Ressourcen zusammen.
- `„resource_max()”` oder `„resource_Maximum()”` oder `„utilization_max()”` oder `„utilization_Maximum()”`:
Liefert die maximale Anzahl an belegten Bedienern in allen Ressourcen zusammen.

12.11 Auslastung der Transporter

12.11.1 Auslastung einer Transportergruppe

- **„transporter_count(id)”**:
Liefert die Anzahl der vorhandenen Transporter in der angegebenen Transportergruppe.
- **„transporter_capacity(id)”**:
Liefert die Kapazität an beförderbaren Kunden eines Transporters in der angegebenen Transportergruppe.
- **„transporter_down(id)”**:
Liefert die Anzahl der momentan in Ausfallzeit befindlichen Transporter in der angegebenen Transportergruppe.
- **„transporter(id)”** oder **„transporter_utilization(id)”**:
Liefert die Anzahl der momentan belegten Transporter in der angegebenen Transportergruppe.
- **„transporter_avg(id)”** oder **„transporter_average(id)”** oder **„transporter_Mittelwert(id)”** oder **„transporter_utilization_avg(id)”** oder **„transporter_utilization_average(id)”** oder **„transporter_utilization_Mittelwert(id)”**:
Liefert die mittlere Anzahl an belegten Transportern in der angegebenen Transportergruppe.
- **„transporter_min(id)”** oder **„transporter_Minimum(id)”** oder **„transporter_utilization_min(id)”** oder **„transporter_utilization_Minimum(id)”**:
Liefert die minimale Anzahl an belegten Transportern in der angegebenen Transportergruppe.
- **„transporter_max(id)”** oder **„transporter_Maximum(id)”** oder **„transporter_utilization_max(id)”** oder **„transporter_utilization_Maximum(id)”**:
Liefert die maximale Anzahl an belegten Transportern in der angegebenen Transportergruppe.
- **„transporter_var(id)”** oder **„transporter_Varianz(id)”** oder **„transporter_utilization_var(id)”** oder **„transporter_utilization_Varianz(id)”**:
Liefert die Varianz der Anzahl an belegten Transportern in der angegebenen Transportergruppe.
- **„transporter_sd(id)”** oder **„transporter_Standardabweichung(id)”** oder **„transporter_utilization_sd(id)”** oder **„transporter_utilization_Standardabweichung(id)”**:
Liefert die Standardabweichung der Anzahl an belegten Transportern in der angegebenen Transportergruppe.
- **„transporter_cv(id)”** oder **„transporter_utilization_cv(id)”**:
Liefert den Variationskoeffizienten der Anzahl an belegten Transportern in der angegebenen Transportergruppe.
- **„transporter_scv(id)”** oder **„transporter_utilization_scv(id)”**:
Liefert den quadrierten Variationskoeffizienten der Anzahl an belegten Transportern in der angegebenen Transportergruppe.
- **„transporter_hist(id;state)”** oder **„transporter_utilization_hist(id;state)”**:
Liefert den Anteil der Zeit, in der **state** Transporter der angegebenen Transportergruppe ausgelastet waren.
- **„transporter_hist(id;stateA;stateB)”** oder **„transporter_utilization_hist(id;stateA;stateB)”**:
Liefert den Anteil der Zeit, mehr als **stateA** und höchstens **stateB** Transporter der angegebenen Transportergruppe ausgelastet waren.

12.11.2 Auslastung aller Transporter zusammen

- **„transporter_count()“:**
Liefert die Anzahl der vorhandenen Transportern in allen Transportergruppen zusammen.
- **„transporter_down()“:**
Liefert die Anzahl der momentan in Ausfallzeit befindlichen Transporter in allen Transportergruppen zusammen.
- **„transporter()“ oder „transporter_utilization()“:**
Liefert die Anzahl der momentan belegten Transporter in allen Transportergruppen zusammen.
- **„transporter_avg()“ oder „transporter_average()“ oder „transporter_Mittelwert()“ oder „transporter_utilization_avg()“ oder „transporter_utilization_average()“ oder „transporter_utilization_Mittelwert()“:**
Liefert die mittlere Anzahl an belegten Transportern in allen Transportergruppen zusammen.
- **„transporter_min()“ oder „transporter_Minimum()“ oder „transporter_utilization_min()“ oder „transporter_utilization_Minimum()“:**
Liefert die minimale Anzahl an belegten Transportern in allen Transportergruppen zusammen.
- **„transporter_max()“ oder „transporter_Maximum()“ oder „transporter_utilization_max()“ oder „transporter_utilization_Maximum()“:**
Liefert die maximale Anzahl an belegten Transportern in allen Transportergruppen zusammen.

12.12 Zugriff auf Statistik-Stationen Datenfelder

- **„Statistik(id;nr)“ oder „Statistics(id;nr)“:**
Liefert den aktuellen Wert des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.
- **„Statistik_avg(id;nr)“ oder „Statistics_avg(id;nr)“ oder „Statistik_Mittelwert(id;nr)“ oder „Statistics_average(id;nr)“:**
Liefert den Mittelwert des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.
- **„Statistik_median(id;nr)“ oder „Statistics_median(id;nr)“:**
Liefert den Median des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.
- **„Statistik_quantil(id;nr;p)“ oder „Statistics_quantil(id;nr;p)“:**
Liefert das Quantil zur Wahrscheinlichkeit **p** des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.
- **„Statistik_min(id;nr)“ oder „Statistics_min(id;nr)“ oder „Statistik_Minimum(id;nr)“ oder „Statistics_Minimum(id;nr)“:**
Liefert den Minimalwert des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.
- **„Statistik_max(id;nr)“ oder „Statistics_max(id;nr)“ oder „Statistik_Maximum(id;nr)“ oder „Statistics_Maximum(id;nr)“:**
Liefert den Maximalwert des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.
- **„Statistik_var(id;nr)“ oder „Statistics_var(id;nr)“ oder „Statistik_Varianz(id;nr)“:**
Liefert die Varianz des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.
- **„Statistik_std(id;nr)“ oder „Statistics_std(id;nr)“ oder „Statistik_Standardabweichung(id;nr)“:**
Liefert die Standardabweichung des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.
- **„Statistik_cv(id;nr)“ oder „Statistics_cv(id;nr)“:**
Liefert den Variationskoeffizienten des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.

- „**Statistik_scv(id;nr)**” oder „**Statistics_scv(id;nr)**”:
Liefert den quadrierten Variationskoeffizienten des Statistikeintrags **nr** (1-basierend) an Statistik-Station **id**.
- „**Statistik_hist(id;nr;state)**” oder „**Statistics_hist(id;nr;state)**”:
Liefert den Anteil der Zeit, in der sich das System in Bezug auf Statistikeintrags **nr** (1-basierend) an Statistik-Station **id** in Zustand **state** befunden hat.
- „**Statistik_hist(id;nr;stateA;stateB)**” oder „**Statistics_hist(id;nr;stateA;stateB)**”:
Liefert den Anteil der Zeit, in der sich das System in Bezug auf Statistikeintrags **nr** (1-basierend) an Statistik-Station **id** in einem Zustand größer als **stateA** und kleiner oder gleich **stateB** befunden hat.

12.13 Zugriff auf Analogwerte

- „**AnalogWert(id)**” oder „**AnalogValue(id)**”:
Liefert den aktuellen Wert des „Analoger Wert”-Elements oder des „Tank”-Elements **id**.
- „**AnalogRate(id)**”:
Liefert die aktuelle Änderungsrate des Wertes des „Analoger Wert”-Elements **id**.
- „**VentilMaximalDurchfluss(id;nr)**” oder „**ValveMaximumFlow(id;nr)**”:
Liefert den aktuellen maximalen Durchfluss an Ventil **nr** (1-basierend) an „Tank”-Element **id**.

12.14 Zugriff auf Kundenobjekt-spezifische Datenfelder

- „**WarmUpKunde()**” oder „**WarmUpClient()**” oder „**isWarmUpClient()**”:
Liefert 0 oder 1 zurück in Abhängigkeit davon, ob der Kunde während der Einschwingphase generiert wurde (1) oder nicht (0).
- „**KundeInStatistik()**” oder „**ClientInStatistics()**” oder „**isClientInStatistics()**”:
Liefert 0 oder 1 zurück in Abhängigkeit davon, ob der Kunde in der Statistik erfasst werden soll (1) oder nicht (0). Der Kunde muss außerdem außerhalb der Einschwingphase generiert worden sein, um tatsächlich erfasst zu werden.
- „**KundeNummer()**” oder „**ClientNumber()**”:
Liefert die 1-basierende fortlaufende Nummer des aktuellen Kunden. Bei der Verwendung von mehreren Simulationsthreads ist diese Zahl thread-lokal.
- „**ClientData(index)**” oder „**KundenDaten(index)**”:
Liefert das an Stelle **index** im aktuellen Kundenobjekt hinterlegte Datenfeld.
In den „Variable”-Elementen kann schreibend auf diese Felder zugegriffen werden.
- „**Alternative()**”:
Gibt an, welche Bedieneralternative an der letzten Bedienstation, die dieser Kunde durchlaufen hat, gewählt wurde. Hat der Kunde noch keine Bedienstation durchlaufen, so liefert die Funktion 0. Ansonsten einen Wert größer oder gleich 1.
- „**PreviousStation()**” oder „**VorherigeStation()**”:
Liefert die ID der Station, an der sich der Kunde vor der aktuellen Station aufgehalten hat.
- „**CurrentWaitingTime()**” oder „**AktuelleWartezeit()**”:
Liefert die bisherige Wartezeit des aktuellen Kunden an der aktuellen Station.

12.15 Zugriff auf die Kosten

- `„costs_waiting_sum(id)“` oder `„Kosten_Wartezeit_Summe(id)“`:
Liefert die Summe der Wartezeitkosten der Kunden, deren Name an Quelle bzw. Namenszuweisung `id` auftritt.
- `„costs_waiting_avg(id)“` oder `„costs_waiting_average(id)“` oder `„Kosten_Wartezeit_Mittelwert(id)“`:
Liefert die durchschnittlichen Wartezeitkosten der Kunden, deren Name an Quelle bzw. Namenszuweisung `id` auftritt.
- `„costs_waiting_sum()“` oder `„Kosten_Wartezeit_Summe()“`:
Liefert die Summe der Wartezeitkosten über alle Kunden.
- `„costs_waiting_avg()“` oder `„costs_waiting_average()“` oder `„Kosten_Wartezeit_Mittelwert()“`:
Liefert die durchschnittlichen Wartezeitkosten über alle Kunden.
- `„costs_waiting()“` oder `„Kosten_Wartezeit()“`:
Liefert die Wartezeitkosten des aktuellen Kunden.
- `„costs_transfer_sum(id)“` oder `„Kosten_Transferzeit_Summe(id)“`:
Liefert die Summe der Transferzeitkosten der Kunden, deren Name an Quelle bzw. Namenszuweisung `id` auftritt.
- `„costs_transfer_avg(id)“` oder `„costs_transfer_average(id)“` oder `„Kosten_Transferzeit_Mittelwert(id)“`:
Liefert die durchschnittlichen Transferzeitkosten der Kunden, deren Name an Quelle bzw. Namenszuweisung `id` auftritt.
- `„costs_transfer_sum()“` oder `„Kosten_Transferzeit_Summe()“`:
Liefert die Summe der Transferzeitkosten über alle Kunden.
- `„costs_transfer_avg()“` oder `„costs_transfer_average()“` oder `„Kosten_Transferzeit_Mittelwert()“`:
Liefert die durchschnittlichen Transferzeitkosten über alle Kunden.
- `„costs_transfer()“` oder `„Kosten_Transferzeit()“`:
Liefert die Transferzeitkosten des aktuellen Kunden.
- `„costs_process_sum(id)“` oder `„Kosten_Bedienzeit_Summe(id)“`:
Liefert die Summe der Bedienzeitkosten der Kunden, deren Name an Quelle bzw. Namenszuweisung `id` auftritt.
- `„costs_process_avg(id)“` oder `„costs_process_average(id)“` oder `„Kosten_Bedienzeit_Mittelwert(id)“`:
Liefert die durchschnittlichen Bedienzeitkosten der Kunden, deren Name an Quelle bzw. Namenszuweisung `id` auftritt.
- `„costs_process_sum()“` oder `„Kosten_Bedienzeit_Summe()“`:
Liefert die Summe der Bedienzeitkosten über alle Kunden.
- `„costs_process_avg()“` oder `„costs_process_average()“` oder `„Kosten_Bedienzeit_Mittelwert()“`:
Liefert die durchschnittlichen Bedienzeitkosten über alle Kunden.
- `„costs_process()“` oder `„Kosten_Bedienzeit()“`:
Liefert die Bedienzeitkosten des aktuellen Kunden.

- **„costs(id)“** oder **„Kosten(id)“**:
Liefert die Stationskosten, die bisher in Summe an Station **id** aufgetreten sind.
- **„costs()“** oder **„Kosten()“**:
Liefert die Stationskosten, die bisher in Summe an allen Stationen aufgetreten sind.
- **„costs_resource(id)“** oder **„Kosten_Ressource(id)“**:
Liefert die Kosten, die durch die angegebene Ressource bisher entstanden sind.
- **„costs_resource()“** oder **„Kosten_Ressource()“**:
Liefert die Kosten, die durch alle Ressourcen bisher entstanden sind.

Kapitel 13

Vergleiche

- „**a == b**“:
Liefert wahr zurück, wenn *a* denselben Wert hat wie *b*.
- „**a != b**“ oder „**a <> b**“:
Liefert wahr zurück, wenn *a* und *b* unterschiedliche Werte aufweisen.
- „**a < b**“:
Liefert wahr zurück, wenn *a* echt kleiner als *b* ist.
- „**a <= b**“ oder „**a =< b**“:
Liefert wahr zurück, wenn *a* echt kleiner oder gleich *b* ist.
- „**a > b**“:
Liefert wahr zurück, wenn *a* echt größer als *b* ist.
- „**a >= b**“ oder „**a => b**“:
Liefert wahr zurück, wenn *a* echt größer oder gleich *b* ist.
- „**A || B**“:
Liefert wahr zurück, wenn *A* oder *B* (oder beide) erfüllt sind.
- „**A && B**“:
Liefert wahr zurück, wenn *A* und *B* erfüllt sind.
- „**!(A)**“:
Liefert wahr zurück, wenn *A* nicht erfüllt ist.

Die Kleinbuchstaben *a* und *b* sind dabei Platzhalter für Rechenausdrücke wie „**WIP()**“. Die Großbuchstaben *A* und *B* stehen für Vergleiche wie „**WIP()<5**“.

13.1 Vergleichsfunktion

Außerdem steht in normalen Rechenausdrücken die „**If**“-Funktion zur Verfügung. Diese erwartet eine ungerade Anzahl an Parametern:

„If(bedingung1;wert1;bedingung2;wert2;...;wertSonst)“

Ist **bedingung1** > 0, so liefert die Funktion **wert1** zurück. Andernfalls prüft sie, ob **bedingung2** > 0 ist und liefert, wenn dies zutrifft, **wert2** zurück usw. Trifft keine der Bedingungen zu, so liefert die Funktion **wertSonst** zurück.

Teil II

Referenz der Javascript-Befehle

An verschiedenen Stellen im Simulator können Skripte verwendet werden. Als Skriptsprachen wird dabei entweder **Javascript** oder **Java** verwendet.

In diesem Abschnitt werden die zusätzlichen **Javascript**-Befehle, die den Zugriff auf die Simulations- oder Statistikdaten ermöglichen und zur Ausgabe der gefilterten Daten zur Verfügung stehen vorgestellt.

Kapitel 14

Statistics-Objekt

Das „**Statistics**“-Objekt ermöglicht den Lesezugriff auf alle Elemente der XML-Datei, die den Statistikdaten zu Grunde liegt. Es ist nur verfügbar, wenn das Skript zum Filtern von Statistikdaten verwendet wird oder innerhalb der Umgebung zur Ausführung von Parameterreihen-Javascripten verwendet wird. Während der Simulation steht das „**Statistics**“-Objekt nicht zur Verfügung. Das Objekt stellt folgende Methoden zur Verfügung:

14.1 Definition des Ausgabeformats

- „**Statistics.setFormat("Format")**“:

Über diesen Befehl kann das Format, in dem „**Statistics.xml**“ Zahlen zur Ausgabe als Zeichenkette formatiert, eingestellt werden. Es kann dabei für Zahlenwerte die lokaler Notation (im deutschsprachigen Raum mit einem Dezimalkomma) oder die System-Notation mit einem Dezimalpunkt ausgegeben werden. Außerdem kann angegeben werden, ob Zahlenwerte als Prozentangabe ausgegeben werden sollen. In diesem Fall wird der Wert mit 100 multipliziert und ein „%“-Zeichen an die Zahl angefügt. Voreingestellt ist stets die Ausgabe in lokaler Notation und die Ausgabe als normale Fließkommazahl (also nicht als Prozentwert).

Folgende Parameter können „**Statistics.setFormat**“ übergeben werden:

- „**System**“: Wahl der System-Notation für Zahlen und Prozentwerte.
- „**Local**“: Wahl der lokalen Notation für Zahlen und Prozentwerte.
- „**Fraction**“: Wahl der Ausgabe als normale Zahl (z.B. 0,357 oder 0.375).
- „**Percent**“: Wahl der Ausgabe als Prozentwert (z.B. 35,7% oder 35.7%).
- „**Time**“: Ausgabe der Zahlenwerte als Zeitangaben (z.B. 00:03:25,87).
- „**Number**“: Ausgabe der Zahlenwerte als normale Zahlen (Ausgabe gemäß Angabe **Percent** oder **Fraction**).

- „**Statistics.setSeparator("Format")**“:

Über diesen Befehl kann eingestellt werden, durch welches Zeichen die einzelnen Einträge einer Verteilung getrennt werden soll, wenn diese über „**Statistics.xml**“ ausgegeben wird. Vorgabe ist die Trennung durch ein Semikolon.

Folgende Parameter können „**Statistics.setSeparator**“ übergeben werden:

- „**Semicolon**“: Semikolons als Trenner

- „Line”: Zeilenumbrüche als Trenner
- „Tabs”: Tabulatoren als Trenner

14.2 Zugriff auf die Statistik-XML-Daten

- **„Statistics.xml("Pfad")”:**

Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde und gibt dieses gemäß den Vorgaben, die per „Statistics.setFormat” und „Statistics.setSeparator” eingestellt wurden, als formatierte Zeichenkette zurück.

Beispiel:

```
var name=Statistics.xml("Modell->ModellName")
```

- **„Statistics.xmlNumber("Pfad")”:**

Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde und gibt den Inhalt als Zahl zurück. Konnte das Feld nicht als Zahlenwert interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

- **„Statistics.xmlArray("Pfad")”:**

Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung und gibt die Werte als Array aus Zahlenwerten zurück. Konnte das Feld nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
Statistics.xmlArray("StatistikBedienzeitenKunden->Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- **„Statistics.xmlSum("Pfad")”:**

Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung, summiert die Werte auf und liefert die Summe als Zahl zurück. Konnte das Feld nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
Statistics.xmlSum("StatistikBedienzeitenKunden->Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- **„Statistics.xmlMean("Pfad")”:**

Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung, bildet den Mittelwert der Werte und liefert diesen als Zahl zurück. Konnte das Feld nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
Statistics.xmlMean("StatistikBedienzeitenKunden->Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- **„Statistics.xmlSD("Pfad")”:**

Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung, bildet die Standardabweichung der Werte und liefert diesen als Zahl zurück. Konnte das Feld nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
Statistics.xmlSD("StatistikBedienzeitenKunden->Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- **„Statistics.xmlCV("Pfad")”:**

Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung, bildet den Variationskoeffizienten der Werte und liefert diesen als Zahl zurück. Konnte das Feld

nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
Statistics.xmlCV("StatistikBedienzeitenKunden->Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- **„Statistics.translate("en")“:**
Übersetzt die Statistikdaten ins Deutsche ("de") oder ins Englische ("en"), so dass jeweils die gewünschten xml-Bezeichner verwendet werden können, auch wenn die Statistikdaten evtl. mit einer anderen Spracheinstellung erstellt wurden.

14.3 Speichern der Statistikdaten in Dateien

- **„Statistics.save("Dateiname")“:**
Speichert die kompletten Statistikdaten in der angegebenen Datei.
Diese Funktion steht nur in der Funktion zur Ausführung von Skripten zur Verfügung.
- **„Statistics.saveNext("Pfad")“:**
Speichert die kompletten Statistikdaten unter dem nächsten verfügbaren Dateinamen in dem angegebenen Verzeichnis.
Diese Funktion steht nur in der Funktion zur Ausführung von Skripten zur Verfügung.
- **„Statistics.filter("Dateiname")“:**
Wendet das angegebene Skript auf die Statistikdaten an und gibt das Ergebnis zurück.
Diese Funktion steht nur in der Funktion zur Ausführung von Skripten zur Verfügung.
- **„Statistics.cancel()“:**
Setzt den Abbruch-Status. (Nach einem Abbruch werden Dateiausgaben nicht mehr ausgeführt.)

14.4 Zugriff auf das Modell

- **„Statistics.getStationID("StationName")“:**
Liefert die ID einer Station basierend auf dem Namen der Station. Existiert keine Station mit dem passenden Namen, so liefert die Funktion -1.

14.5 Abfrage der zugehörigen Statistikdatei

- **„Statistics.getStatisticsFile()“:**
Liefert den vollständigen Pfad- und Dateinamen der Statistikdatei, aus der die Daten stammen. Stammen die Statistikdaten nicht aus einer Datei, so wird eine leere Zeichenkette zurück geliefert.
- **„Statistics.getStatisticsFileName()“:**
Liefert den Dateinamen der Statistikdatei, aus der die Daten stammen. Stammen die Statistikdaten nicht aus einer Datei, so wird eine leere Zeichenkette zurück geliefert.

Kapitel 15

System-Objekt

Das „**System**“-Objekt ermöglicht den Zugriff auf einige allgemeine Programmfunktionen. Es ist nur verfügbar, wenn das Skript zum Filtern von Statistikdaten verwendet wird oder die Funktion zur Ausführung von Parameterreihen-Javascripten aktiv ist. Während der Simulation steht das „**System**“-Objekt nicht zur Verfügung. Das Objekt stellt folgende Methoden zur Verfügung:

- **„System.calc("Ausdruck")“:**
Berechnet den als Zeichenkette übergebenen Ausdruck mit Hilfe der Termauswertungsfunktion, die auch an verschiedenen anderen Stellen im Programm zur Anwendung kommt (siehe Teil I) und liefert das Ergebnis als Zahl zurück. Konnte der Ausdruck nicht berechnet werden, so wird eine Fehlermeldung als Zeichenkette zurückgeliefert. Die Termauswertung ermöglicht den Zugriff auf alle bekannten Wahrscheinlichkeitsverteilungen, den Erlang-C-Rechner usw.
- **„System.time()“:**
Liefert die aktuelle Systemzeit als Millisekunden-Wert zurück. Diese Funktion kann zur Messung der Laufzeit des Skriptes verwendet werden.
- **„System.getInput("http://Adresse",-1)“:**
Lädt einen Zahlenwert über die angegebene Adresse und liefert diesen zurück. Wenn kein Wert geladen werden konnte, wird der im zweiten Parameter angegebene Fehlerwert zurückgeliefert.
- **„System.execute("program.exe")“:**
Führt ein externes Programm aus und kehrt sofort zurück. Liefert true, wenn das Programm gestartet werden konnte. Die Ausführung externer Programme durch Skripte ist standardmäßig deaktiviert und muss zunächst im Einstellungen-Dialog aktiviert.
- **„System.executeAndReturnOutput("program.exe")“:**
Führt ein externes Programm aus und liefert die Ausgaben des Programms zurück. Die Ausführung externer Programme durch Skripte ist standardmäßig deaktiviert und muss zunächst im Einstellungen-Dialog aktiviert.
- **„System.executeAndWait("program.exe")“:**
Führt ein externes Programm aus und liefert den Rückgabecode des Programms zurück. Im Fehlerfall wird -1 zurückgeliefert. Die Ausführung externer Programme durch Skripte ist standardmäßig deaktiviert und muss zunächst im Einstellungen-Dialog aktiviert.

Kapitel 16

Simulation-Objekt

Das „**Simulation**“-Objekt ermöglicht den Zugriff auf die aktuellen Simulationsdaten während der Laufzeit der Simulation. Es ist nur verfügbar während die Simulation läuft und kann bei der späteren Filterung der Ergebnisse nicht verwendet werden. Das Objekt stellt folgende Methoden zur Verfügung:

16.1 Basisfunktionen

- **„Simulation.time()“**:
Liefert die aktuelle Zeit in der Simulation als Sekunden-Zahlenwert.
- **„Simulation.calc("Ausdruck")“**:
Berechnet den als Zeichenkette übergebenen Ausdruck mit Hilfe der Termauswertungsfunktion, die auch an verschiedenen anderen Stellen im Programm zur Anwendung kommt (siehe Teil I) und liefert das Ergebnis als Zahl zurück. Konnte der Ausdruck nicht berechnet werden, so wird eine Fehlermeldung als Zeichenkette zurückgeliefert. Die Termauswertung ermöglicht den Zugriff auf alle bekannten Wahrscheinlichkeitsverteilungen, den Erlang-C-Rechner usw.
- **„Simulation.getInput("http://Adresse",-1)“**:
Lädt einen Zahlenwert über die angegebene Adresse und liefert diesen zurück. Wenn kein Wert geladen werden konnte, wird der im zweiten Parameter angegebene Fehlerwert zurückgeliefert.
- **„Simulation.execute("program.exe")“**:
Führt ein externes Programm aus und kehrt sofort zurück. Liefert true, wenn das Programm gestartet werden konnte. Die Ausführung externer Programme durch Skripte ist standardmäßig deaktiviert und muss zunächst im Einstellungen-Dialog aktiviert.
- **„Simulation.executeAndReturnOutput("program.exe")“**:
Führt ein externes Programm aus und liefert die Ausgaben des Programms zurück. Die Ausführung externer Programme durch Skripte ist standardmäßig deaktiviert und muss zunächst im Einstellungen-Dialog aktiviert.
- **„Simulation.executeAndWait("program.exe")“**:
Führt ein externes Programm aus und liefert den Rückgabecode des Programms zurück. Im Fehlerfall wird -1 zurückgeliefert. Die Ausführung externer Programme durch Skripte ist standardmäßig deaktiviert und muss zunächst im Einstellungen-Dialog aktiviert.
- **„Simulation.isWarmUp()“**:
Liefert wahr oder falsch zurück in Abhängigkeit, ob sich die Simulation noch in der Einschwingphase befindet.

- **„Simulation.getMapLocal()“:**
Liefert eine stations-lokale Zuordnung, in die Werte geschrieben und aus der Werte gelesen werden können. Die hier gespeicherten Werte bleiben über die Ausführung des aktuellen Skriptes hinaus erhalten.
- **„Simulation.getMapGlobal()“:**
Liefert eine modellweite Zuordnung, in die Werte geschrieben und aus der Werte gelesen werden können. Die hier gespeicherten Werte bleiben über die Ausführung des aktuellen Skriptes hinaus erhalten.
- **„Simulation.terminateSimulation(message)“:**
Beendet die Simulation. Wird als Nachricht **null** übergeben, so wird die Simulation normal beendet. Im Falle einer Nachricht wird die Simulation mit der entsprechenden Fehlermeldung abgebrochen.

16.2 Zugriff auf kundenspezifische Daten

- **„Simulation.clientTypeName()“:**
Liefert den Namen des Typs des Kunden, der die Verarbeitung des Skripts ausgelöst hat.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.isWarmUpClient()“:**
Liefert wahr oder falsch zurück in Abhängigkeit, ob der Kunde während der Einschwingphase generiert wurde und daher nicht in der Statistik erfasst werden soll.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.isClientInStatistics()“:**
Liefert wahr oder falsch zurück in Abhängigkeit, ob der Kunde in der Statistik erfasst werden soll. Diese Einstellung ist unabhängig von der Einschwingphase. Ein Kunde wird nur erfasst, wenn er außerhalb der Einschwingphase generiert wurde und hier nicht falsch zurückgeliefert wird.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.setClientInStatistics(inStatistics)“:**
Stellt ein, ob ein Kunde in der Statistik erfasst werden soll. Diese Einstellung ist unabhängig von der Einschwingphase. Ein Kunde wird nur erfasst, wenn er außerhalb der Einschwingphase generiert wurde und hier nicht falsch eingestellt wurde.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientNumber()“:**
Liefert die bei 1 beginnende, fortlaufende Nummer des aktuellen Kunden. Werden mehrere Simulationsthreads verwendet, so ist dieser Wert thread-lokal.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientWaitingSeconds()“:**
Liefert die bisherige Wartezeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als Sekunden-Zahlenwert zurück.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientWaitingTime()“:**
Liefert die bisherige Wartezeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als formatierte Zeitangabe als String zurück.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientWaitingSecondsSet(seconds)“:**
Stellt die bisherige Wartezeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, ein.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)

- **„Simulation.clientTransferSeconds()“**:
Liefert die bisherige Transferzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als Sekunden-Zahlenwert zurück.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientTransferTime()“**:
Liefert die bisherige Transferzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als formatierte Zeitangabe als String zurück.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientTransferSecondsSet(seconds)“**:
Stellt die bisherige Transferzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, ein.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientProcessSeconds()“**:
Liefert die bisherige Bedienzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als Sekunden-Zahlenwert zurück.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientProcessTime()“**:
Liefert die bisherige Bedienzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als formatierte Zeitangabe als String zurück.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientProcessSecondsSet(seconds)“**:
Stellt die bisherige Bedienzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, ein.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientResidenceSeconds()“**:
Liefert die bisherige Verweilzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als Sekunden-Zahlenwert zurück.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientResidenceTime()“**:
Liefert die bisherige Verweilzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als formatierte Zeitangabe als String zurück.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.clientResidenceSecondsSet(seconds)“**:
Stellt die bisherige Verweilzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, ein.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.getClientValue(Index)“**:
Liefert den zu dem **Index** für den aktuellen Kunden hinterlegten Zahlenwert.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.setClientValue(Index,Wert)“**:
Stellt für den aktuellen Kunden für **Index** den Zahlenwert **Wert** ein.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.getClientText("Schlüssel")“**:
Liefert die zu **Schlüssel** für den aktuellen Kunden hinterlegte Zeichenkette.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)
- **„Simulation.setClientText("Schlüssel","Wert")“**:
Stellt für den aktuellen Kunden für **Schlüssel** die Zeichenkette **Wert** ein.
(Sofern das Ereignis direkt durch einen Kunden ausgelöst wurde.)

16.3 Temporäre Batche

Handelt es sich bei dem aktuellen Kunden um einen temporären Batch, so kann auf die Eigenschaften der in ihm enthaltenen inneren Kunden lesend zugegriffen werden:

- **„Simulation.batchSize()“**:
Liefert die Anzahl an Kunden, die sich in dem temporären Batch befinden. Ist der aktuelle Kunden keine temporärer Batch, so liefert die Funktion 0.
- **„Simulation.getBatchTypeName(batchIndex)“**:
Liefert den Namen eines der Kunden in dem aktuellen Batch. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchWaitingSeconds(batchIndex)“**:
Liefert die bisherige Wartezeit eines der Kunden in dem aktuellen Batch in Sekunden als Zahlenwert. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchWaitingTime(batchIndex)“**:
Liefert die bisherige Wartezeit eines der Kunden in dem aktuellen Batch in formatierter Form als Zeichenkette. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchTransferSeconds(batchIndex)“**:
Liefert die bisherige Transferzeit eines der Kunden in dem aktuellen Batch in Sekunden als Zahlenwert. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchTransferTime(batchIndex)“**:
Liefert die bisherige Transferzeit eines der Kunden in dem aktuellen Batch in formatierter Form als Zeichenkette. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchProcessSeconds(batchIndex)“**:
Liefert die bisherige Bedienzeit eines der Kunden in dem aktuellen Batch in Sekunden als Zahlenwert. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchProcessTime(batchIndex)“**:
Liefert die bisherige Bedienzeit eines der Kunden in dem aktuellen Batch in formatierter Form als Zeichenkette. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchResidenceSeconds(batchIndex)“**:
Liefert die bisherige Verweilzeit eines der Kunden in dem aktuellen Batch in Sekunden als Zahlenwert. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchResidenceTime(batchIndex)“**:
Liefert die bisherige Verweilzeit eines der Kunden in dem aktuellen Batch in formatierter Form als Zeichenkette. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchValue(batchIndex, index)“**:
Liefert einen zu einem der Kunden in dem aktuellen Batch einen gespeicherten Zahlenwert. Der übergebene Batch-Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„Simulation.getBatchText(batchIndex, key)“**:
Liefert zu einem der Kunden in dem aktuellen Batch einen gespeicherten Textwert. Der übergebene Batch-Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.

16.4 Zugriff auf Parameter des Simulationsmodells

- **„Simulation.set("Name",Wert)“:**
Setzt die Simulationsvariable, deren Name im ersten Parameter angegeben wurde auf den im zweiten Parameter angegebenen Wert. **Wert** kann dabei eine Zahl oder eine Zeichenkette sein. Im Falle einer Zahl erfolgt eine direkte Zuweisung. Zeichenketten werden gemäß **Simulation.calc** interpretiert und das Ergebnis an die Variable zugewiesen. Bei **Name** muss es sich um entweder eine an anderer Stelle definierte Simulationsvariable handeln oder um ein Kundendaten-Feld der Form **ClientData(index)** mit *index* ≥ 0 .
- **„Simulation.setValue(id,Wert)“:**
Stellt den Wert an dem „Analoger Wert“- oder „Tank“-Element mit der angegebenen **id** ein.
- **„Simulation.setRate(id,Wert)“:**
Stellt die Änderungsrate (pro Sekunde) an dem „Analoger Wert“-Element mit der angegebenen **id** ein.
- **„Simulation.setValveMaxFlow(id,VentilNr,Wert)“:**
Stellt den maximalen Durchfluss (pro Sekunde) an dem angegebenen Ventil (1-basierend) des „Tank“-Elements mit der angegebenen **id** ein. Der maximale Durchfluss muss dabei eine nichtnegative Zahl sein.
- **„Simulation.getWIP(id)“:**
Liefert die aktuelle Anzahl an Kunden an der Station mit der angegebenen Id.
- **„Simulation.getNQ(id)“:**
Liefert die aktuelle Anzahl an Kunden in der Warteschlange an der Station mit der angegebenen Id.
- **„Simulation.getWIP()“:**
Liefert die aktuelle Anzahl an Kunden im System.
- **„Simulation.getNQ()“:**
Liefert die aktuelle Anzahl an im System wartenden Kunden.

16.5 Zugriff auf den aktuellen Eingabewert

- **„Simulation.getInput()“:**
Erfolgt die Javascript-Verarbeitung innerhalb eines „Eingabe (Skript)“-Elements, so kann über diese Funktion der aktuelle Eingabewert abgerufen werden. Andernfalls liefert diese Funktion lediglich den Wert 0.

16.6 Anzahl an Bedienern in einer Ressource

- **„Simulation.getAllResourceCount()“:**
Liefert die aktuelle Anzahl an Bedienern in allen Ressourcen zusammen.
- **„Simulation.getResourceCount(id)“:**
Liefert die aktuelle Anzahl an Bedienern in der Ressource mit der angegebenen Id.
- **„Simulation.setResourceCount(id,count)“:**
Stellt die Anzahl an Bedienern an der Ressource mit der angegebenen Id ein. Damit die Anzahl an Bedienern in der Ressource zur Laufzeit verändert werden kann, muss initial eine feste Anzahl an Bedienern (nicht unendliche viele und nicht über einen Zeitplan) in der Ressource definiert sein. Außerdem dürfen keine Ausfälle für die Ressource eingestellt sein. Die Funktion liefert **true** zurück, wenn die Anzahl an Bedienern erfolgreich geändert werden konnte. Wenn der neue Wert geringer als

der bisherige Wert ist, so ist der neue Wert evtl. nicht sofort im Simulationssystem ersichtlich, da eigentlich nicht mehr vorhandene Bediener zunächst aktuelle Bedienungen zu Ende führen, bevor diese entfernt werden.

- `„Simulation.getAllResourceDown()“`:
Liefert die aktuelle Anzahl an Bedienern in Ausfallzeit über alle Ressourcen.
- `„Simulation.getResourceDown(id)“`:
Liefert die aktuelle Anzahl an Bedienern in Ausfallzeit in der Ressource mit der angegebenen Id.

16.7 Signale auslösen

- `„Simulation.signal(name)“`:
Löst das Signal mit dem angegebenen Namen aus.

16.8 Meldung in Logging ausgeben

- `„Simulation.log(message)“`:
Gibt die übergebene Meldung im Logging-System aus (sofern eine Logaufzeichnung aktiviert ist).

16.9 Kunden an Verzögerungen-Stationen freigeben

Wurde an einer Verzögerung-Station eingestellt, dass eine Liste der aktuell dort befindlichen Kunden mitgeführt werden soll, so kann diese Liste über die folgende Funktion abgefragt werden und es können selektiv einzelne Kunden vor Ablauf ihrer angegebenen Verzögerungszeit freigegeben werden.

- `„getDelayStationData(id)“`:
Liefert ein Objekt, welches die unter **Zugriff auf kundenspezifische Daten** angegebenen Methoden zur Abfrage der Liste der Kunden an der Verzögerungs-Station **id** bereitstellt. Ist die ID ungültig so wird **null** zurückgeliefert.

Kapitel 17

Clients-Objekt

Das „**Clients**“-Objekt steht nur innerhalb des Skript-Bedingung-Elements zur Verfügung und hält alle Informationen zu den wartenden Kunden vor. Des Weiteren ermöglicht es, einzelne Kunden freizugeben.

- **„Clients.count()“**:
Liefert die Anzahl an wartenden Kunden. Bei den anderen Methode kann über den Index-Parameter (Wert 0 bis **count()**-1) auf einen bestimmten Kunden zugegriffen werden.
- **„Clients.clientTypeName(index)“**:
Liefert den Namen des Typs des Kunden.
- **„Clients.clientWaitingSeconds(index)“**:
Liefert die bisherige Wartezeit des Kunden als Sekunden-Zahlenwert zurück.
- **„Clients.clientWaitingTime(index)“**:
Liefert die bisherige Wartezeit des Kunden als formatierte Zeitangabe als String zurück.
- **„Clients.clientTransferSeconds(index)“**:
Liefert die bisherige Transferzeit des Kunden als Sekunden-Zahlenwert zurück.
- **„Clients.clientTransferTime(index)“**:
Liefert die bisherige Transferzeit des Kunden als formatierte Zeitangabe als String zurück.
- **„Clients.clientProcessSeconds(index)“**:
Liefert die bisherige Bedienzeit des Kunden als Sekunden-Zahlenwert zurück.
- **„Clients.clientProcessTime(index)“**:
Liefert die bisherige Bedienzeit des Kunden als formatierte Zeitangabe als String zurück.
- **„Clients.clientResidenceSeconds(index)“**:
Liefert die bisherige Verweilzeit des Kunden als Sekunden-Zahlenwert zurück.
- **„Clients.clientResidenceTime(index)“**:
Liefert die bisherige Verweilzeit des Kunden als formatierte Zeitangabe als String zurück.
- **„Clients.clientData(index,data)“**:
Liefert das über den zweiten Parameter adressierte Datum des angegebenen Kunden zurück.
- **„Clients.clientData(index,data,value)“**:
Stellt den als dritten Parameter übergebenen Zahlenwert an der als zweiten Parameter adressierten Kundendatenposition des angegebenen Kunden ein.
- **„Clients.clientTextData(index,key)“**:
Liefert den Wert es über den zweiten Parameter adressierte Schlüssels des angegebenen Kunden zurück.

- **„Clients.clientTextData(index,key,value)”**:
Stellt den als dritten Parameter übergebenen Wert für den als zweiten Parameter adressierten Schlüssel des angegebenen Kunden ein.
- **„Clients.release(index)”**:
Veranlasst die Weiterleitung des angegebenen Kunden.

Kapitel 18

Output-Objekt

Das „**Output**“-Objekt stellt Funktionen zur Ausgabe der gefilterten Ergebnisse zur Verfügung:

- **„Output.setFormat("Format")“:**

Über diesen Befehl kann das Format, in dem „**Output.print**“ und „**Output.println**“ Zahlen formatieren, eingestellt werden. Es kann dabei für Zahlenwerte die lokaler Notation (im deutschsprachigen Raum mit einem Dezimalkomma) oder die System-Notation mit einem Dezimalpunkt ausgegeben werden. Außerdem kann angegeben werden, ob Zahlenwerte als Prozentangabe ausgegeben werden sollen. In diesem Fall wird der Wert mit 100 multipliziert und ein „%“ Zeichen an die Zahl angefügt. Voreingestellt ist stets die Ausgabe in lokaler Notation und die Ausgabe als normale Fließkommazahl (also nicht als Prozentwert).

Folgende Parameter können **Output.setFormat** übergeben werden:

- **„System“:**
Wahl der System-Notation für Zahlen und Prozentwerte.
- **„Local“:**
Wahl der lokalen Notation für Zahlen und Prozentwerte.
- **„Fraction“:**
Wahl der Ausgabe als normale Zahl (z.B. 0,357 oder 0.375).
- **„Percent“:**
Wahl der Ausgabe als Prozentwert (z.B. 35,7% oder 35.7%).
- **„Number“:**
Interpretation von Zahlenwerten als normale Zahlen (Dezimalwert oder Prozentwert).
- **„Time“:**
Interpretation von Zahlenwerten als Zeitangaben.

- **„Output.setSeparator("Format")“:**

Über diesen Befehl kann eingestellt werden, durch welches Zeichen die einzelnen Einträge eines Arrays getrennt werden soll, wenn diese über „**Output.print**“ oder „**Output.println**“ ausgegeben werden. Vorgabe ist die Trennung durch ein Semikolon.

Folgende Parameter können **Output.setSeparator** übergeben werden:

- **„Semicolon“:**
Semikolons als Trenner.

- **„Line“**:
Zeilenumbrüche als Trenner.
- **„Tabs“**:
Tabulatoren als Trenner.
- **„Output.setDigits(digits)“**:
Über diesen Befehl kann eingestellt werden, wie viele Nachkommastellen bei der Ausgabe von Zahlen gemäß lokaler Notation ausgegeben werden sollen. Ein negativer Wert bedeutet, dass alle verfügbaren Nachkommastellen ausgegeben. (Bei Verwendung der System-Notation werden stets alle verfügbaren Nachkommastellen ausgegeben.)
- **„Output.print("Ausdruck")“**:
Gibt den übergebenen Ausdruck aus. Zeichenketten werden direkt ausgegeben. Zahlenwerte werden gemäß den per **Output.setFormat** vorgenommenen Einstellungen formatiert.
- **„Output.println("Ausdruck")“**:
Gibt den übergebenen Ausdruck aus und fügt dabei einen Zeilenumbruch an. Zeichenketten werden direkt ausgegeben. Zahlenwerte werden gemäß den per **Output.setFormat** vorgenommenen Einstellungen formatiert.
- **„Output.newLine()“**:
Gibt einen Zeilenumbruch aus. Diese Funktion ist gleichwertig zu dem Aufruf von **„Output.println(“”)“**.
- **„Output.tab()“**:
Gibt einen Tabulator aus.
- **„Output.cancel()“**:
Setzt den Abbruch-Status. (Nach einem Abbruch werden Dateiausgaben nicht mehr ausgeführt.)
- **„Output.printlnDDE("Arbeitsmappe","Tabelle","Zelle","Ausdruck")“**:
Dieser Befehl steht nur zur Verfügung, wenn DDE verfügbar ist, d.h. unter Windows. Er gibt den übergebenen Ausdruck per DDE in der angegebenen Tabelle in Excel aus. Zahlenwerte werden dabei gemäß den per **Output.setFormat** vorgenommenen Einstellungen formatiert.

Kapitel 19

FileOutput-Objekt

Das „**FileOutput**“-Objekt stellt alle Funktionen, die auch das „**Output**“-Objekt anbietet, zur Verfügung und ist nur während der Parameterreihen-Skript-Ausführung verfügbar. Im Unterschied zum „**Output**“-Objekt werden die Ausgaben nicht auf die Standardausgabe geleitet, sondern es muss zunächst per „**FileOutput.setFile("Dateiname")**“ eine Ausgabedatei definiert werden. Alle Ausgaben werden dann an diese Datei angehängt.

Kapitel 20

Model-Objekt

Das „**Model**“-Objekt steht nur während der Parameterreihen-Javascript-Ausführung zur Verfügung und bietet Funktionen, um auf Modelleigenschaften zuzugreifen und Simulationen zu initiieren.

- **„Model.reset()“:**
Stellt das Modell auf den Ausgangszustand zurück.
- **„Model.run()“:**
Simuliert das aktuelle Modell. Auf die Ergebnisse kann im Folgenden über das „**Statistics**“-Objekt zugegriffen werden.
- **„Model.setDistributionParameter("Pfad", Index, Zahl)“:**
Stellt einen Verteilungsparameter **Index** (zwischen 1 und 4) der über **Pfad** angegebenen Wahrscheinlichkeitsverteilung ein.
- **„Model.setMean("Pfad", Zahl)“:**
Stellt den Mittelwert der über **"Pfad"** angegebenen Wahrscheinlichkeitsverteilung auf den angegebenen Wert.
- **„Model.setSD("Pfad", Zahl)“:**
Stellt die Standardabweichung der über **Pfad** angegebenen Wahrscheinlichkeitsverteilung auf den angegebenen Wert.
- **„Model.setString("Pfad", "Text")“:**
Schreibt an die über **Pfad** angegebene Stelle im Modell die angegebene Zeichenkette.
- **„Model.setValue("Pfad", Zahl)“:**
Schreibt an die über **Pfad** angegebene Stelle im Modell den angegebenen Wert.
- **„Model.xml("Pfad")“:**
Liefert den über **Pfad** erreichbaren Wert.
Diese Funktion ist das Äquivalent zu **„Statistics.xml("Pfad")“** für Modelldaten.
- **„Model.getResourceCount("RessourcenName")“:**
Liefert die Anzahl an Bedienern in der Ressource mit Namen **RessourcenName**. Existiert die Ressource nicht oder ist in ihr die Bedieneranzahl nicht als Zahlenwert hinterlegt, so liefert die Funktion -1. Ansonsten die Anzahl an Bedienern in der Ressource.
- **„Model.setResourceCount("RessourcenName", Anzahl)“:**
Stellt die Anzahl an Bedienern in der Ressource mit Namen **RessourcenName** ein.
- **„Model.getGlobalVariableInitialValue("VariablenName")“:**
Liefert den Ausdruck zur Bestimmung des initialen Wertes der globalen Variable **VariablenName**. Existiert die globale Variable nicht, so wird eine leere Zeichenkette geliefert.

- `„Model.setGlobalVariableInitialValue("VariablenName", "Ausdruck")“`:
Stellt den Ausdruck zur Bestimmung des initialen Wertes der globalen Variable **VariablenName** ein.
- `„Model.cancel()“`:
Setzt den Abbruch-Status. (Nach einem Abbruch werden keine Simulationen mehr ausgeführt.)
- `„Model.getStationID("StationName")“`:
Liefert die ID einer Station basierend auf dem Namen der Station. Existiert keine Station mit dem passenden Namen, so liefert die Funktion -1.

Kapitel 21

XML-Auswahlbefehle

Über die Parameter der Funktionen des „**Statistics**“-Objektes kann der Inhalt eines XML-Elements oder der Wert eines Attributes eines XML-Elements ausgelesen werden. Die Selektion eines XML-Elements erfolgt dabei mehrstufig getrennt durch `->` Zeichen. Zwischen den `->` Zeichen stehen jeweils die Namen von XML-Elementen. Zusätzlich können in eckigen Klammern Namen und Werte von Attributen angegeben werden, nach denen gefiltert werden soll.

Beispiele:

- `„Statistics.xml("Modell->ModellName")“`:
Liefert den Inhalt des Elements **ModellName**, welches ein Unterelement von **Modell** ist.
- `„Statistics.xml("StatistikZwischenankunftszeitenKunden->Station[Typ=\"Quelle id=1\"]->[Mittelwert]")“`:
Selektiert das **Station**-Unterelement des **StatistikZwischenankunftszeitenKunden**-Elements, bei dem das **Typ**-Attribut auf den Wert **Quelle id=1** gesetzt ist. Und liefert dann den Inhalt des Attributs **Mittelwert**.

Teil III

Referenz der Java-Befehle

An verschiedenen Stellen im Simulator können Skripte verwendet werden. Als Skriptsprachen wird dabei entweder **Javascript** oder **Java** verwendet.

In diesem Abschnitt werden die zusätzlichen **Java**-Befehle, die den Zugriff auf die Simulations- oder Statistikdaten ermöglichen und zur Ausgabe der gefilterten Daten zur Verfügung stehen vorgestellt.

Der **Java**-Code muss in eine Methode der Form

```
void function(SimulationInterface sim) {  
  
}
```

eingebettet werden. Neben den Standardsprachbefehlen kann abhängig vom Kontext, in dem das Skript ausgeführt wird, über die Methoden des übergebenen **SimulationInterface** weitere Interfaces, die ihrerseits weitere Methoden mitbringen, auf die Simulations- oder Statistikdaten zugegriffen werden:

Kapitel 22

StatisticsInterface abrufbar über `sim.getStatistics()`

Das über `sim.getStatistics()` gelieferte `StatisticsInterface`-Interface ermöglicht den Lesezugriff auf alle Elemente der XML-Datei, die den Statistikdaten zu Grunde liegt. Es ist nur verfügbar, wenn das Skript zum Filtern von Statistikdaten verwendet wird oder innerhalb der Umgebung zur Ausführung von Parameterreihen-Skripten verwendet wird. Während der Simulation liefert `sim.getStatistics()` lediglich `null`. Das Interface stellt folgende Methoden zur Verfügung:

22.1 Definition des Ausgabeformats

- `„void setFormat(final String format)“`:

Über diesen Befehl kann das Format, in dem `„Statistics.xml“` Zahlen zur Ausgabe als Zeichenkette formatiert, eingestellt werden. Es kann dabei für Zahlenwerte die lokaler Notation (im deutschsprachigen Raum mit einem Dezimalkomma) oder die System-Notation mit einem Dezimalpunkt ausgegeben werden. Außerdem kann angegeben werden, ob Zahlenwerte als Prozentangabe ausgegeben werden sollen. In diesem Fall wird der Wert mit 100 multipliziert und ein `„%“`-Zeichen an die Zahl angefügt. Voreingestellt ist stets die Ausgabe in lokaler Notation und die Ausgabe als normale Fließkommazahl (also nicht als Prozentwert). Folgende Parameter können `„Statistics.setFormat“` übergeben werden:

- `„System“`: Wahl der System-Notation für Zahlen und Prozentwerte.
- `„Local“`: Wahl der lokalen Notation für Zahlen und Prozentwerte.
- `„Fraction“`: Wahl der Ausgabe als normale Zahl (z.B. 0,357 oder 0.375).
- `„Percent“`: Wahl der Ausgabe als Prozentwert (z.B. 35,7% oder 35.7%).
- `„Time“`: Ausgabe der Zahlenwerte als Zeitangaben (z.B. 00:03:25,87).
- `„Number“`: Ausgabe der Zahlenwerte als normale Zahlen (Ausgabe gemäß Angabe `Percent` oder `Fraction`).

- `„void setSeparator(final String separator)“`:

Über diesen Befehl kann eingestellt werden, durch welches Zeichen die einzelnen Einträge einer Verteilung getrennt werden soll, wenn diese über `„Statistics.xml“` ausgegeben wird. Vorgabe ist die Trennung durch ein Semikolon. Folgende Parameter können `„Statistics.setSeparator“` übergeben werden:

- `„Semicolon“`: Semikolons als Trenner
- `„Line“`: Zeilenumbrüche als Trenner

- „Tabs”: Tabulatoren als Trenner

22.2 Zugriff auf die Statistik-XML-Daten

- „`String xml(final String path)`”:
Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde und gibt dies gemäß den Vorgaben, die per `sim.getStatistics().setFormat` und `sim.getStatistics().setSeparator` eingestellt wurden, als formatierte Zeichenkette zurück.

Beispiel: `String name=sim.getStatistics().xml("Modell->ModellName")`

- „`Object xmlNumber(final String path)`”:
Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde und gibt den Inhalt als **Double**-Zahl zurück. Konnte das Feld nicht als Zahlenwert interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.
- „`Object xmlArray(final String path)`”:
Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung und gibt die Werte als Array aus Zahlenwerten (**double[]**) zurück. Konnte das Feld nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
sim.getStatistics().xmlArray("StatistikBedienzeitenKunden->
Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- „`Object xmlSum(final String path)`”:
Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung, summiert die Werte auf und liefert die Summe als **Double**-Zahl zurück. Konnte das Feld nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
sim.getStatistics().xmlSum("StatistikBedienzeitenKunden->
Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- „`Object xmlMean(final String path)`”:
Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung, bildet den Mittelwert der Werte und liefert diesen als **Double**-Zahl zurück. Konnte das Feld nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
sim.getStatistics().xmlMean("StatistikBedienzeitenKunden->
Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- „`Object xmlSD(final String path)`”:
Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung, bildet die Standardabweichung der Werte und liefert diesen als **Double**-Zahl zurück. Konnte das Feld nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
sim.getStatistics().xmlSD("StatistikBedienzeitenKunden->
Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- „`Object xmlCV(final String path)`”:
Lädt das XML-Datenfeld, dessen Pfad als Parameter angegeben wurde, interpretiert dieses als Verteilung, bildet den Variationskoeffizienten der Werte und liefert diesen als **Double**-Zahl zurück. Konnte

das Feld nicht als Verteilung interpretiert werden, so wird eine Zeichenkette mit einer Fehlermeldung zurückgegeben.

Beispiel:

```
sim.getStatistics().xmlCV("StatistikBedienzeitenKunden->Kundentyp[Typ=\"KundenA\"]->[Verteilung]")
```

- **„boolean translate(final String language)“:**
Übersetzt die Statistikdaten ins Deutsche ("de") oder ins Englische ("en"), so dass jeweils die gewünschten xml-Bezeichner verwendet werden können, auch wenn die Statistikdaten evtl. mit einer anderen Spracheinstellung erstellt wurden.

22.3 Speichern der Statistikdaten in Dateien

- **„boolean save(final String fileName)“:**
Speichert die kompletten Statistikdaten in der angegebenen Datei.
Diese Funktion steht nur in der Funktion zur Ausführung von Skripten zur Verfügung.
- **„boolean saveNext(final String folderName)“:**
Speichert die kompletten Statistikdaten unter dem nächsten verfügbaren Dateinamen in dem angegebenen Verzeichnis.
Diese Funktion steht nur in der Funktion zur Ausführung von Skripten zur Verfügung.
- **„String filter(final String fileName)“:**
Wendet das angegebene Skript auf die Statistikdaten an und gibt das Ergebnis zurück.
Diese Funktion steht nur in der Funktion zur Ausführung von Skripten zur Verfügung.
- **„void cancel()“:**
Setzt den Abbruch-Status. (Nach einem Abbruch werden Dateiausgaben nicht mehr ausgeführt.)

22.4 Zugriff auf das Modell

- **„int getStationID(final String name)“:**
Liefert die ID einer Station basierend auf dem Namen der Station. Existiert keine Station mit dem passenden Namen, so liefert die Funktion -1.

22.5 Abfrage der zugehörigen Statistikdatei

- **„String getStatisticsFile()“:**
Liefert den vollständigen Pfad- und Dateinamen der Statistikdatei, aus der die Daten stammen. Stammen die Statistikdaten nicht aus einer Datei, so wird eine leere Zeichenkette zurück geliefert.
- **„String getStatisticsFileName()“:**
Liefert den Dateinamen der Statistikdatei, aus der die Daten stammen. Stammen die Statistikdaten nicht aus einer Datei, so wird eine leere Zeichenkette zurück geliefert.

Kapitel 23

RuntimeInterface abrufbar über `sim.getRuntime`

Das `RuntimeInterface`-Interface ermöglicht den Zugriff auf einige allgemeine Programmfunktionen. Es ist immer verfügbar. Das Interface stellt folgende Methoden zur Verfügung:

- **„Object calc(final String expression)“:**
Berechnet den als Zeichenkette übergebenen Ausdruck mit Hilfe der Termauswertungsfunktion, die auch an verschiedenen anderen Stellen im Programm zur Anwendung kommt (siehe Teil I) und liefert das Ergebnis als **Double**-Zahl zurück. Konnte der Ausdruck nicht berechnet werden, so wird eine Fehlermeldung als Zeichenkette zurückgeliefert. Die Termauswertung ermöglicht den Zugriff auf alle bekannten Wahrscheinlichkeitsverteilungen, den Erlang-C-Rechner usw.
- **„long getTime()“:**
Liefert die aktuelle Systemzeit als Millisekunden-Wert zurück. Diese Funktion kann zur Messung der Laufzeit des Skriptes verwendet werden.
- **„double getInput(final String url, final double errorValue)“:**
Lädt einen Zahlenwert über die angegebene Adresse und liefert diesen zurück. Wenn kein Wert geladen werden konnte, wird der im zweiten Parameter angegebene Fehlerwert zurückgeliefert.
- **„boolean execute(final String commandLine)“:**
Führt ein externes Programm aus und kehrt sofort zurück. Liefert true, wenn das Programm gestartet werden konnte. Die Ausführung externer Programme durch Skripte ist standardmäßig deaktiviert und muss zunächst im Einstellungen-Dialog aktiviert.
- **„String executeAndReturnOutput(final String commandLine)“:**
Führt ein externes Programm aus und liefert die Ausgaben des Programms zurück. Die Ausführung externer Programme durch Skripte ist standardmäßig deaktiviert und muss zunächst im Einstellungen-Dialog aktiviert.
- **„int executeAndWait(final String commandLine)“:**
Führt ein externes Programm aus und liefert den Rückgabecode des Programms zurück. Im Fehlerfall wird -1 zurückgeliefert. Die Ausführung externer Programme durch Skripte ist standardmäßig deaktiviert und muss zunächst im Einstellungen-Dialog aktiviert.

Kapitel 24

SystemInterface abrufbar über `sim.getSystem()`

Das `SystemInterface`-Interface ermöglicht den Zugriff auf die aktuellen Simulationsdaten während der Laufzeit der Simulation. Es ist nur verfügbar während die Simulation läuft und kann bei der späteren Filterung der Ergebnisse nicht verwendet werden. Das Interface stellt folgende Methoden zur Verfügung:

24.1 Basisfunktionen

- `„double getTime()“`:
Liefert die aktuelle Zeit in der Simulation als Sekunden-Zahlenwert.
- `„Object calc(final String expression)“`:
Berechnet den als Zeichenkette übergebenen Ausdruck mit Hilfe der Termauswertungsfunktion, die auch an verschiedenen anderen Stellen im Programm zur Anwendung kommt (siehe Teil I) und liefert das Ergebnis als `Double`-Zahl zurück. Konnte der Ausdruck nicht berechnet werden, so wird eine Fehlermeldung als Zeichenkette zurückgeliefert. Die Termauswertung ermöglicht den Zugriff auf alle bekannten Wahrscheinlichkeitsverteilungen, den Erlang-C-Rechner usw.
- `„boolean isWarmUp()“`:
Liefert wahr oder falsch zurück in Abhängigkeit, ob sich die Simulation noch in der Einschwingphase befindet.
- `„Map<String,Object> getMapLocal()“`:
Liefert eine stations-lokale Zuordnung, in die Werte geschrieben und aus der Werte gelesen werden können. Die hier gespeicherten Werte bleiben über die Ausführung des aktuellen Skriptes hinaus erhalten.
- `„Map<String,Object> getMapGlobal()“`:
Liefert eine modellweite Zuordnung, in die Werte geschrieben und aus der Werte gelesen werden können. Die hier gespeicherten Werte bleiben über die Ausführung des aktuellen Skriptes hinaus erhalten.
- `„void terminateSimulation(final String message)“`:
Terminates the simulation. If `null` is passed as message, the simulation is terminated normally. In case of a message, the simulation will be terminated with the corresponding error message.

24.2 Zugriff auf Parameter des Simulationsmodells

- `„void set(final String varName, final Object varValue)“`:
Setzt die Simulationsvariable, deren Name im ersten Parameter angegeben wurde auf den im zweiten Parameter angegebenen Wert. `varValue` kann dabei eine Zahl oder eine Zeichenkette sein. Im

Falle einer Zahl erfolgt eine direkte Zuweisung. Zeichenketten werden gemäß `calc(final String expression)` interpretiert und das Ergebnis an die Variable zugewiesen. Bei `varName` muss es sich um entweder eine an anderer Stelle definierte Simulationsvariable handeln oder um ein Kundendaten-Feld der Form `ClientData(index)` mit $index \geq 0$.

- `„void setAnalogValue(final Object elementID, final Object value)“`:
Stellt den Wert an dem „Analoger Wert“- oder „Tank“-Element mit der angegebenen Id ein.
- `„void setAnalogRate(final Object elementID, final Object value)“`:
Stellt die Änderungsrate (pro Sekunde) an dem „Analoger Wert“-Element mit der angegebenen Id ein.
- `„void setAnalogValveMaxFlow(final Object elementID, final Object valveNr, final Object value)“`:
Stellt den maximalen Durchfluss (pro Sekunde) an dem angegebenen Ventil (1-basierend) des „Tank“-Elements mit der angegebenen Id ein. Der maximale Durchfluss muss dabei eine nichtnegative Zahl sein.
- `„int getWIP(final int id)“`:
Liefert die aktuelle Anzahl an Kunden an der Station mit der angegebenen Id.
- `„int getNQ(final int id)“`:
Liefert die aktuelle Anzahl an Kunden in der Warteschlange an der Station mit der angegebenen Id.
- `„int getWIP()“`:
Liefert die aktuelle Anzahl an Kunden im System.
- `„int getNQ()“`:
Liefert die aktuelle Anzahl an im System wartenden Kunden.

24.3 Anzahl an Bedienern in einer Ressource

- `„int getAllResourceCount()“`:
Liefert die aktuelle Anzahl an Bedienern in allen Ressourcen zusammen.
- `„int getResourceCount(final int resourceId)“`:
Liefert die aktuelle Anzahl an Bedienern in der Ressource mit der angegebenen Id.
- `„boolean setResourceCount(final int resourceId, final int count)“`:
Stellt die Anzahl an Bedienern an der Ressource mit der angegebenen Id ein. Damit die Anzahl an Bedienern in der Ressource zur Laufzeit verändert werden kann, muss initial eine feste Anzahl an Bedienern (nicht unendliche viele und nicht über einen Zeitplan) in der Ressource definiert sein. Außerdem dürfen keine Ausfälle für die Ressource eingestellt sein. Die Funktion liefert `true` zurück, wenn die Anzahl an Bedienern erfolgreich geändert werden konnte. Wenn der neue Wert geringer als der bisherige Wert ist, so ist der neue Wert evtl. nicht sofort im Simulationssystem ersichtlich, da eigentlich nicht mehr vorhandene Bediener zunächst aktuelle Bedienungen zu Ende führen, bevor diese entfernt werden.
- `„int getAllResourceDown()“`:
Liefert die aktuelle Anzahl an Bedienern in Ausfallzeit über alle Ressourcen.
- `„int getResourceDown(final int resourceId)“`:
Liefert die aktuelle Anzahl an Bedienern in Ausfallzeit in der Ressource mit der angegebenen Id.

24.4 Signale auslösen

- „`signal(final String signalName)`“:
Löst das Signal mit dem angegebenen Namen aus.

24.5 Externen Code aufrufen

- „`Object runPlugin(final String className, final String functionName, final Object data)`“:
Ruft in der angegebenen Klasse die angegebene Methode auf und übermittelt an diese den in **data** angegebenen optionalen Parameter. Der Rückgabewert der Methode wird von **runPlugin** zurückgegeben. Schlägt der Aufruf fehl, so liefert **runPlugin** den Wert **null** zurück.

24.6 Meldung in Logging ausgeben

- „`void log(final Object obj)`“:
Gibt die übergebene Meldung im Logging-System aus (sofern eine Logaufzeichnung aktiviert ist).

24.7 Kunden an Verzögerungen-Stationen freigegeben

Wurde an einer Verzögerung-Station eingestellt, dass eine Liste der aktuell dort befindlichen Kunden mitgeführt werden soll, so kann diese Liste über die folgende Funktion abgefragt werden und es können selektiv einzelne Kunden vor Ablauf ihrer angegebenen Verzögerungszeit freigegeben werden.

- „`ClientsInterface getDelayStationData(final int id)`“:
Liefert ein Objekt, welches das Interface **ClientsInterface** implementiert und die Liste der Kunden an Verzögerungs-Station **id** repräsentiert. Ist die ID ungültig so wird **null** zurückgeliefert.

Kapitel 25

ClientInterface abrufbar über `sim.getClient()`

Das `ClientInterface`-Interface ermöglicht den Zugriff auf die Simulationsdaten des aktuellen Kunden während der Laufzeit der Simulation. Es ist nur verfügbar während die Simulation läuft die Verarbeitung durch einen Kunden ausgelöst wurde. Das Interface stellt folgende Methoden zur Verfügung:

- **„Object calc(final String expression)“:**
Berechnet den als Zeichenkette übergebenen Ausdruck mit Hilfe der Termauswertungsfunktion, die auch an verschiedenen anderen Stellen im Programm zur Anwendung kommt (siehe Teil I) und liefert das Ergebnis als **Double**-Zahl zurück. Konnte der Ausdruck nicht berechnet werden, so wird eine Fehlermeldung als Zeichenkette zurückgeliefert. Die Termauswertung ermöglicht den Zugriff auf alle bekannten Wahrscheinlichkeitsverteilungen, den Erlang-C-Rechner usw.
- **„String getTypeName()“:**
Liefert den Namen des Typs des Kunden, der die Verarbeitung des Skripts ausgelöst hat.
- **„boolean isWarmUp()“:**
Liefert wahr oder falsch zurück in Abhängigkeit, ob der Kunde während der Einschwingphase generiert wurde und daher nicht in der Statistik erfasst werden soll.
- **„boolean isInStatistics()“:**
Liefert wahr oder falsch zurück in Abhängigkeit, ob der Kunde in der Statistik erfasst werden soll. Diese Einstellung ist unabhängig von der Einschwingphase. Ein Kunde wird nur erfasst, wenn er außerhalb der Einschwingphase generiert wurde und hier nicht falsch zurückgeliefert wird.
- **„void setInStatistics(final boolean inStatistics)“:**
Stellt ein, ob ein Kunde in der Statistik erfasst werden soll. Diese Einstellung ist unabhängig von der Einschwingphase. Ein Kunde wird nur erfasst, wenn er außerhalb der Einschwingphase generiert wurde und hier nicht falsch eingestellt wurde.
- **„long getNumber()“:**
Liefert die bei 1 beginnende, fortlaufende Nummer des aktuellen Kunden. Werden mehrere Simulationsthreads verwendet, so ist dieser Wert Thread-lokal.
- **„double getWaitingSeconds()“:**
Liefert die bisherige Wartezeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als Sekunden-Zahlenwert zurück.
- **„String getWaitingTime()“:**
Liefert die bisherige Wartezeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als formatierte Zeitangabe als String zurück.
- **„void setWaitingSeconds(final double seconds)“:**
Stellt die bisherige Wartezeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, ein.

- **„double getTransferSeconds()“**:
Liefert die bisherige Transferzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als Sekunden-Zahlenwert zurück.
- **„String getTransferTime()“**:
Liefert die bisherige Transferzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als formatierte Zeitangabe als String zurück.
- **„void setTransferSeconds(final double seconds)“**:
Stellt die bisherige Transferzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, ein.
- **„double getProcessSeconds()“**:
Liefert die bisherige Bedienzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als Sekunden-Zahlenwert zurück.
- **„String getProcessTime()“**:
Liefert die bisherige Bedienzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als formatierte Zeitangabe als String zurück.
- **„void setProcessSeconds(final double seconds)“**:
Stellt die bisherige Bedienzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, ein.
- **„double getResidenceSeconds()“**:
Liefert die bisherige Verweilzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als Sekunden-Zahlenwert zurück.
- **„String getResidenceTime()“**:
Liefert die bisherige Verweilzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, als formatierte Zeitangabe als String zurück.
- **„void setResidenceSeconds(final double seconds)“**:
Stellt die bisherige Verweilzeit des Kunden, der die Verarbeitung des Skripts ausgelöst hat, ein.
- **„double getValue(final int index)“**:
Liefert den zu dem `index` für den aktuellen Kunden hinterlegten Zahlenwert.
- **„void setValue(final int index, final int value)“,
„void setValue(final int index, final double value)“,
„void setValue(final int index, final String value)“**:
Stellt für den aktuellen Kunden für `index` den Wert `value` ein. Wird als `value` eine Zeichenkette übergeben, so wird diese zunächst über die `calc(final String expression)`-Funktion ausgewertet.
- **„String getText(final String key)“**:
Liefert die zu `key` für den aktuellen Kunden hinterlegte Zeichenkette.
- **„void setText(final String key, final String value)“**:
Stellt für den aktuellen Kunden für `key` die Zeichenkette `value` ein.

25.1 Temporäre Batche

Handelt es sich bei dem aktuellen Kunden um einen temporären Batch, so kann auf die Eigenschaften der in ihm enthaltenen inneren Kunden lesend zugegriffen werden:

- **„int batchSize()“**:
Liefert die Anzahl an Kunden, die sich in dem temporären Batch befinden. Ist der aktuelle Kunden keine temporärer Batch, so liefert die Funktion 0.

- **„String getBatchTypeName(final int batchIndex)“:**
Liefert den Namen eines der Kunden in dem aktuellen Batch. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„double getBatchWaitingSeconds(final int batchIndex)“:**
Liefert die bisherige Wartezeit eines der Kunden in dem aktuellen Batch in Sekunden als Zahlenwert. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„String getBatchWaitingTime(final int batchIndex)“:**
Liefert die bisherige Wartezeit eines der Kunden in dem aktuellen Batch in formatierter Form als Zeichenkette. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„double getBatchTransferSeconds(final int batchIndex)“:**
Liefert die bisherige Transferzeit eines der Kunden in dem aktuellen Batch in Sekunden als Zahlenwert. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„String getBatchTransferTime(final int batchIndex)“:**
Liefert die bisherige Transferzeit eines der Kunden in dem aktuellen Batch in formatierter Form als Zeichenkette. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„double getBatchProcessSeconds(final int batchIndex)“:**
Liefert die bisherige Bedienzeit eines der Kunden in dem aktuellen Batch in Sekunden als Zahlenwert. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„String getBatchProcessTime(final int batchIndex)“:**
Liefert die bisherige Bedienzeit eines der Kunden in dem aktuellen Batch in formatierter Form als Zeichenkette. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„double getBatchResidenceSeconds(final int batchIndex)“:**
Liefert die bisherige Verweilzeit eines der Kunden in dem aktuellen Batch in Sekunden als Zahlenwert. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„String getBatchResidenceTime(final int batchIndex)“:**
Liefert die bisherige Verweilzeit eines der Kunden in dem aktuellen Batch in formatierter Form als Zeichenkette. Der übergebene Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„double getBatchValue(final int batchIndex, final int index)“:**
Liefert einen zu einem der Kunden in dem aktuellen Batch einen gespeicherten Zahlenwert. Der übergebene Batch-Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.
- **„String getBatchText(final int batchIndex, final String key)“:**
Liefert zu einem der Kunden in dem aktuellen Batch einen gespeicherten Textwert. Der übergebene Batch-Index ist 0-basierend und muss im Bereich von 0 bis **batchSize()-1** liegen.

Kapitel 26

InputValueInterface abrufbar über `sim.getInputValue()`

Das `InputValueInterface`-Interface ermöglicht das Abrufen des nächsten Eingabewertes, sofern die Skript-Verarbeitung innerhalb eines Eingabe (Skript)-Elements angestoßen wurde. Das Interface stellt folgende Methode zur Verfügung:

- **double get():**
Über diese Funktion kann der aktuelle Eingabewert abgerufen werden.

Kapitel 27

ClientsInterface abrufbar über `sim.getClients()`

Das `ClientsInterface`-Interface steht nur innerhalb des Skript-Bedingung-Elements zur Verfügung und hält alle Informationen zu den wartenden Kunden vor. Des Weiteren ermöglicht es, einzelne Kunden freizugeben.

- **„int count()“:**
Liefert die Anzahl an wartenden Kunden. Bei den anderen Methode kann über den Index-Parameter (Wert 0 bis `count()-1`) auf einen bestimmten Kunden zugegriffen werden.
- **„String clientTypeName(final int index)“:**
Liefert den Namen des Typs des Kunden.
- **„double clientWaitingSeconds(final int index)“:**
Liefert die bisherige Wartezeit des Kunden als Sekunden-Zahlenwert zurück.
- **„String clientWaitingTime(final int index)“:**
Liefert die bisherige Wartezeit des Kunden als formatierte Zeitangabe als String zurück.
- **„double clientTransferSeconds(final int index)“:**
Liefert die bisherige Transferzeit des Kunden als Sekunden-Zahlenwert zurück.
- **„String clientTransferTime(final int index)“:**
Liefert die bisherige Transferzeit des Kunden als formatierte Zeitangabe als String zurück.
- **„double clientProcessSeconds(final int index)“:**
Liefert die bisherige Bedienzeit des Kunden als Sekunden-Zahlenwert zurück.
- **„String clientProcessTime(final int index)“:**
Liefert die bisherige Bedienzeit des Kunden als formatierte Zeitangabe als String zurück.
- **„double clientResidenceSeconds(final int index)“:**
Liefert die bisherige Verweilzeit des Kunden als Sekunden-Zahlenwert zurück.
- **„String clientResidenceTime(final int index)“:**
Liefert die bisherige Verweilzeit des Kunden als formatierte Zeitangabe als String zurück.
- **„double clientData(final int index, final int data)“:**
Liefert das über den zweiten Parameter adressierte Datum des angegebenen Kunden zurück.
- **„void clientData(final int index, final int data, final double value)“:**
Stellt den als dritten Parameter übergebenen Zahlenwert an der als zweiten Parameter adressierten Kundendatenposition des angegebenen Kunden ein.
- **„String clientTextData(final int index, final String key)“:**
Liefert den Wert es über den zweiten Parameter adressierte Schlüssels des angegebenen Kunden zurück.

- **„String clientTextData(final int index, final String key, final String value)“:**
Stellt den als dritten Parameter übergebenen Wert für den als zweiten Parameter adressierten Schlüssel des angegebenen Kunden ein.
- **„void release(final int index)“:**
Veranlasst die Weiterleitung des angegebenen Kunden.

Kapitel 28

OutputInterface abrufbar über `sim.getOutputStream()`

Das `OutputInterface`-Interface stellt Funktionen zur Ausgabe der gefilterten Ergebnisse zur Verfügung:

- `„void setFormat(final String format)“`:

Über diesen Befehl kann das Format, in dem `print` und `println` Zahlen formatieren, eingestellt werden. Es kann dabei für Zahlenwerte die lokaler Notation (im deutschsprachigen Raum mit einem Dezimalkomma) oder die System-Notation mit einem Dezimalpunkt ausgegeben werden. Außerdem kann angegeben werden, ob Zahlenwerte als Prozentangabe ausgegeben werden sollen. In diesem Fall wird der Wert mit 100 multipliziert und ein `%` Zeichen an die Zahl angefügt. Voreingestellt ist stets die Ausgabe in lokaler Notation und die Ausgabe als normale Fließkommazahl (also nicht als Prozentwert).

Folgende Parameter können `setFormat` übergeben werden:

- `„System“`:
Wahl der System-Notation für Zahlen und Prozentwerte.
- `„Local“`:
Wahl der lokalen Notation für Zahlen und Prozentwerte.
- `„Fraction“`:
Wahl der Ausgabe als normale Zahl (z.B. 0,357 oder 0.375).
- `„Percent“`:
Wahl der Ausgabe als Prozentwert (z.B. 35,7% oder 35.7%).
- `„Number“`:
Interpretation von Zahlenwerten als normale Zahlen (Dezimalwert oder Prozentwert).
- `„Time“`:
Interpretation von Zahlenwerten als Zeitangaben.

- `„void setSeparator(final String separator)“`:

Über diesen Befehl kann eingestellt werden, durch welches Zeichen die einzelnen Einträge eines Arrays getrennt werden soll, wenn diese über `print` oder `println` ausgegeben werden. Vorgabe ist die Trennung durch ein Semikolon.

Folgende Parameter können `setSeparator` übergeben werden:

- `„Semicolon“`:
Semikolons als Trenner.
- `„Line“`:
Zeilenumbrüche als Trenner.

- „**Tabs**“:
Tabulatoren als Trenner.
- „**void setDigits(final int digits)**“:
Über diesen Befehl kann eingestellt werden, wie viele Nachkommastellen bei der Ausgabe von Zahlen gemäß lokaler Notation ausgegeben werden sollen. Ein negativer Wert bedeutet, dass alle verfügbaren Nachkommastellen ausgegeben. (Bei Verwendung der System-Notation werden stets alle verfügbaren Nachkommastellen ausgegeben.)
- „**void print(final Object obj)**“:
Gibt den übergebenen Ausdruck aus. Zeichenketten werden direkt ausgegeben. Zahlenwerte werden gemäß den per **setFormat** vorgenommenen Einstellungen formatiert.
- „**void println(final Object obj)**“:
Gibt den übergebenen Ausdruck aus und fügt dabei einen Zeilenumbruch an. Zeichenketten werden direkt ausgegeben. Zahlenwerte werden gemäß den per **setFormat** vorgenommenen Einstellungen formatiert.
- „**void newLine()**“:
Gibt einen Zeilenumbruch aus. Diese Funktion ist gleichwertig zu dem Aufruf von **println("")**.
- „**void tab()**“:
Gibt einen Tabulator aus.
- „**void cancel()**“:
Setzt den Abbruch-Status. (Nach einem Abbruch werden Dateiausgaben nicht mehr ausgeführt.)
- „**printlnDDE(final String workbook, final String table, final String cell, final Object obj)**“:
Dieser Befehl steht nur zur Verfügung, wenn DDE verfügbar ist, d.h. unter Windows. Er gibt den übergebenen Ausdruck per DDE in der angegebenen Tabelle in Excel aus. Zahlenwerte werden dabei gemäß den per **setFormat** vorgenommenen Einstellungen formatiert.

Kapitel 29

FileOutputInterface abrufbar über `sim.getFileOutput()`

Das **FileOutputInterface**-Interface stellt alle Funktionen, die auch das **OutputInterface**-Interface anbietet, zur Verfügung und ist nur während der Parameterreihen-Skript-Ausführung verfügbar. Im Unterschied zum **OutputInterface**-Interface werden die Ausgaben nicht auf die Standardausgabe geleitet, sondern es muss zunächst per `sim.getFileOutput().setFile("Dateiname")` eine Ausgabedatei definiert werden. Alle Ausgaben werden dann an diese Datei angehängt.

Kapitel 30

ModelInterface abrufbar über `sim.getModel()`

Das `ModelInterface`-Interface steht nur während der Parameterreihen-Skript-Ausführung zur Verfügung und bietet Funktionen, um auf Modelleigenschaften zuzugreifen und Simulationen zu initiieren.

- `void reset()`:
Stellt das Modell auf den Ausgangszustand zurück.
- `void run()`:
Simuliert das aktuelle Modell. Auf die Ergebnisse kann im Folgenden über das `<tt>StatisticsInterface</tt>`-Interface zugegriffen werden.
- `boolean setDistributionParameter(final String xmlName, final int number, final double value)`:
Stellt einen Verteilungsparameter `number` (zwischen 1 und 4) der über `xmlName` angegebenen Wahrscheinlichkeitsverteilung ein.
- `boolean setMean(final String xmlName, final double value)`:
Stellt den Mittelwert der über `xmlName` angegebenen Wahrscheinlichkeitsverteilung auf den angegebenen Wert.
- `boolean setSD(final String xmlName, final double value)`:
Stellt die Standardabweichung der über `xmlName` angegebenen Wahrscheinlichkeitsverteilung auf den angegebenen Wert.
- `boolean setString(final String xmlName, final String value)`:
Schreibt an die über `xmlName` angegebene Stelle im Modell die angegebene Zeichenkette.
- `boolean setValue(final String xmlName, final double value)`:
Schreibt an die über `xmlName` angegebene Stelle im Modell den angegebenen Wert.
- `String xml(final String xmlName)`:
Liefert den über `xmlName` erreichbaren Wert.
Diese Funktion ist das Äquivalent zu `sim.getStatistics().xml(xmlName)` für Modelldaten.
- `getResourceCount(final String resourceName)`:
Liefert die Anzahl an Bedienern in der Ressource mit Namen `resourceName`. Existiert die Ressource nicht oder ist in ihr die Bedieneranzahl nicht als Zahlenwert hinterlegt, so liefert die Funktion -1. Ansonsten die Anzahl an Bedienern in der Ressource.
- `boolean setResourceCount(final String resourceName, final int count)`:
Stellt die Anzahl an Bedienern in der Ressource mit Namen `resourceName` ein.

- **„String getGlobalVariableInitialValue(final String variableName)“:**
Liefert den Ausdruck zur Bestimmung des initialen Wertes der globalen Variable mit Namen **variableName**. Existiert die globale Variable nicht, so wird eine leere Zeichenkette geliefert.
- **„boolean setGlobalVariableInitialValue(final String variableName, final String expression)“:**
Stellt den Ausdruck zur Bestimmung des initialen Wertes der globalen Variable mit Namen **variableName** ein.
- **„void cancel()“:**
Setzt den Abbruch-Status. (Nach einem Abbruch werden keine Simulationen mehr ausgeführt.)
- **„int getStationID(final String name)“:**
Liefert die ID einer Station basierend auf dem Namen der Station. Existiert keine Station mit dem passenden Namen, so liefert die Funktion -1.

Kapitel 31

XML-Auswahlbefehle

Über die Parameter der Funktionen des „**StatisticsInterface**“-Interfaces kann der Inhalt eines XML-Elements oder der Wert eines Attributes eines XML-Elements ausgelesen werden. Die Selektion eines XML-Elements erfolgt dabei mehrstufig getrennt durch "->Zeichen. Zwischen den "->Zeichen stehen jeweils die Namen von XML-Elementen. Zusätzlich können in eckigen Klammern Namen und Werte von Attributen angegeben werden, nach denen gefiltert werden soll.

Beispiele:

- `„sim.getStatistics().xml("Modell->ModellName")“`:
Liefert den Inhalt des Elements **ModellName**, welches ein Unterelement von **Modell** ist.
- `„sim.getStatistics().xml("StatistikZwischenankunftszeitenKunden->Station[Typ=\"Quelle id=1\"]->[Mittelwert]")“`:
Selektiert das **Station**-Unterelement des **StatistikZwischenankunftszeitenKunden**-Elements, bei dem das **Typ**-Attribut auf den Wert **Quelle id=1** gesetzt ist. Und liefert dann den Inhalt des Attributs **Mittelwert**.