OESTERREICHISCHE NATIONALBANK
EUROSYSTEM

DV-technische Schnittstelle

Prüfregelsyntax für Meldungen und Rückmeldungen

Oesterreichische Nationalbank

VERSION 3.00

Versionsübersicht

Letzter Review: 16.06.2020

Version	Ersteller/ Verantwortlicher	Erstell-/ Änderungsdatum	Prüfer/ Freigeber
1.0	Sledz Markus	12.11.2013	Cordt Christoph
1.1	Sledz Markus	-	
1.2	Sledz Markus	-	
1.3	Sledz Markus	09.03.2015	Cordt Christoph
1.4	Sledz Markus	26.06.2015	Cordt Christoph
1.5	Sledz Markus	12.10.2015	Cordt Christoph
1.6	Sledz Markus	03.10.2016	Cordt Christoph
1.7	Sledz Markus	02.02.2017	Baar Lukas
1.8	Sledz Markus	20.10.2017	Baar Lukas
1.9	Sledz Markus	30.10.2017	Baar Lukas
1.10	Sledz Markus	27.12.2017	Baar Lukas
1.11	Sledz Markus	01.03.2018	Baar Lukas
1.12	Sledz Markus	09.07.2018	Baar Lukas
1.13	Sledz Markus	22.08.2018	Baar Lukas
1.14	Sledz Markus	09.11.2018	Baar Lukas
1.15	Sledz Markus	11.12.2018	Baar Lukas
1.16	Sledz Markus	11.07.2019	Baar Lukas
1.17	Sledz Markus	28.10.2019	Baar Lukas
2.0	Mikl Christina	18.05.2020	Sledz Markus
2.1	Mikl Christina	05.01.2021	Sledz Markus
2.2	Neubauer-Stiedl Klemens	27.09.2021	Sledz Markus
2.3	Neubauer-Stiedl Klemens	02.12.2021	Sledz Markus

2.4	Neubauer-Stiedl Klemens	02.02.2022	Sledz Markus
2.5	Neubauer-Stiedl Klemens	14.04.2022	Sledz Markus
2.6	Neubauer-Stiedl Klemens	23.06.2022	Sledz Markus
2.7	Neubauer-Stiedl Klemens	01.08.2022	Sledz Markus
2.8	Neubauer-Stiedl Klemens	17.01.2023	Sledz Markus
2.9	Neubauer-Stiedl Klemens	07.02.2023	Sledz Markus
2.10	Neubauer-Stiedl Klemens	16.02.2023	Sledz Markus
2.11	Neubauer-Stiedl Klemens	03.01.2024	-
3.00	Imre Thomas, Neubauer-Stiedl Klemens, Thrakl Clemens	28.03.2024	-

Version 1.0:	DV-Schnittstelle für die Prüfregelsyntax von ein- und mehrdimensionalen Erhebungen, sowie Smart Cubes
Version 1.1:	Anpassungen in Kapitel 3.1; Korrekturen in Funktionentabelle - einheitliche Schreibweise in der Spalte „Beispiel“
Version 1.2:	Neues Kapitel 2.1 „Konzepte“, Neues Kapitel 2.2 „Rechenregeln“, Neues Kapitel 3.3 „Beispiel“, Anpassungen der Formatierung
Version 1.3:	Erweiterung Kapitel 1.1 „Grundlagen“, Erweiterung Kapitel 1.2 „Logische Werte / Exceptional Value“, Erweiterung Kapitel 3.4 „Beispiel“
Version 1.4:	Erweiterung um Beschreibung der Operatoren und des Vektorraums
Version 1.5:	Neues Kapitel „Dimensionsvergleich“ und „Gruppierungen“
Version 1.6:	Erweiterung Kapitel 3.1 „Funktionen“
Version 1.7:	Erweiterung Kapitel 1.1 „Grundlagen“, Erweiterung Kapitel 2 „Bestandteile einer Rechenregel“, Anpassung Kapitel 1.3 „Arithmetische Operationen“, Neues Unterkapitel „Dimension Meldeperiode (MP)“, Erweiterung Kapitel 3.1 „Funktionen“, Neues Unterkapitel „Reservierte Schlüsselwörter“, Anpassungen Kapitel 4 „Ansprechpersonen“, Anpassung der Versionsübersicht
Version 1.8:	Erweiterung Kapitel 1.6 „Rechnen mit Vektoren“, Erweiterung Kapitel 2 „Bestandteile einer Rechenregel“, Neues Unterkapitel „Stammdatenfunktionen“
Version 1.9:	Erweiterung Kapitel 3.4 „Stammdatenfunktionen“
Version 1.10:	Erweiterung Kapitel 3.4 „Stammdatenfunktionen“

	Einstellung Kapitel 3.5 „Rückfrage-Beispiel“
Version 1.11	Erweiterung der Beschreibung zur Funktion „crossjoin“
	Erweiterung Kapitel 3.4 „Stammdatenfunktionen“
Version 1.12	Erweiterung Kapitel 3.4 „Stammdatenfunktionen“
	Neues Unterkapitel „Implementierung“
	Anpassung Kapitel 1 „Allgemein“
Version 1.13	Neues Unterkapitel „Stringfunktionen“
Version 1.14	Erweiterung Unterkapitel „Relationale Operatoren“
	Erweiterung Unterkapitel „Aggregatsfunktion“
	Erweiterung Funktion aggInc und aggExc
Version 1.15	Neues Unterkapitel im Kapitel „Funktionen“
Version 1.16	Neues Unterkapitel „Syntax-Kommentar“
	Umbenennung von errorDiv0 in errDiv
	Neue Funktion pruefFremdschluessel
	Redaktionelle Erweiterungen Funktion getAttribute
	Erweiterung von multDiffVector um Datentypkompatibilität
	Anpassungen der Formatierung
	Redaktionelle Anpassungen
Version 1.17	Neue Funktionen dateDiff, EoMonth, asDate
	Entfernung von Funktionen anyTrue, anyFalse, allTrue, allFalse
Version 2.0	Redaktionelle Anpassungen
Version 2.1	Neue Funktionen unaryMinus, intersection, union, currentDate, checkBez, getHoechste, getMeldePflicht, getMeldetermin, getNeuInsolvenz, pruefWKN
	Entfernung von Funktionen getRatingOeNBG, reklassifikation
	Verwendung von geschwungenen Klammern {,} bei _G
	Aggregatsfunktion - Median /Z
Version 2.2	Neue Funktion matchRegEx

	Erweiterungen Funktionen getAttribute, checkBez
	Neues Unterkapitel „OeNB-interne Funktionen“
	Ergänzung Beispiel im Unterkapitel „Aggregationsfunktionen“
	Erweiterung der Ansprechpartner
	Redaktionelle Anpassungen
Version 2.3	Redaktionelle Anpassungen bei den Funktionen aggInc und aggExc
Version 2.4	Erweiterung des Unter-Kapitels ignoreNa / ignoreNaN
	Korrektur der Schreibweise der Funktionen errDiv und errDivNeg
	Korrektur der Tabelle für fixe Meldeperioden (Kapitel 2.3.4)
	Korrektur eines Beispiels bei aggExc
	Erweiterung der Funktionen isNa, currentDate, replace
	Neues Unterkapitel „Mengenfunktionen“
	Neue Funktionen difference, symmDifference
	Redaktionelle Anpassungen
Version 2.5	Kapitel 1.2. beim Ergebnis „(Fehlerprotokoll)“ entfernt
	Entfernen von ignoreNaN (wird in Prüfungen nicht verwendet)
	Unterkapitel ignoreNa: Entfernen von mean und median; detailliertere Beispiele für sum, max, min ergänzt
	Arithmetische Operatoren: „+ Verleiht einer Variable ein positives Vorzeichen“ entfernt
	Arithmetische Operatoren: Ergänzung der äquivalenten Funktionen
	Ergänzung der Kapitel-Überschriften „Datentypprüfungen“ und „Konvertierungs-Funktionen“ und Verschiebung der entsprechenden Funktionen
	unaryNot Information bezüglich cast von numerischen Werten ergänzt
	aggExc: Korrektur der Algorithmus-Beschreibung
	aggExc: Korrektur des Ergebnisses bei Beispiel 2
	Erweiterung der Funktionen isChar, asChar, isLogical, asLogical
	getBezAttribut: Ergänzung Umgang #NR wie mit NA
	mean & median: Beispiel mit ausschließlich NAs ergänzt (Kapitel 3)
	Redaktionelle Anpassungen
Version 2.6	interne OeNB Funktionen aus der Tabelle am Anfang des Kapitels Funktionen nach OeNB interne Funktionen verschoben
	Textanpassung im Kapitel Umgang mit NA (Unterkapitel von „String-Funktionen“)
	Erweiterung der Funktionen isNumeric, asNumeric, length

	Korrektur von aggInc & aggExc in der Datenkompatibilitätstabelle (NA & Leere Menge)
	Ergänzung von Halbjahr bei den variablen Meldeperiodenangaben
	Redaktionelle Anpassungen
Version 2.7	Erweiterung der Funktionen mapping, and, or, unaryNot
	Redaktionelle Anpassungen
Version 2.8	Erweiterung der Funktionen xor, xnor, pow
	Verschiebung der Funktionen errDiv und pow von den OeNB-internen Funktionen nach Funktionen
	Erweiterung der replace-Funktion: Umgang mit NA as String
	and-, or-Funktion: Ergänzung von Beispielen mit leerer Menge (Beispiele 3 bis 6)
	Entfernen der Funktionen getMeldePflicht und getMeldetermin
	Kapitel ignoreNa: Ergänzung von aggExc und aggInc
	aggExc, aggInc: Ergänzung in der Datenkompatibilitätstabelle für NA mit ignoreNa
	mapping: Ergänzung in der Beschreibung zum Umgang mit NA, Korrektur der Testfälle und Ergänzung von Testfällen
	Redaktionelle Anpassungen
Version 2.9	Erweiterung der Funktionen unaryMinus, isApprox, allUnique
	pow: Ergänzung von Beispielen
	Redaktionelle Anpassungen
Version 2.10	mapping: Ergänzung in der Beschreibung, Korrektur in den „Weiteren Beispielen“
Version 2.11	Verschiebung der Funktion contains von OeNB-internen Funktionen nach Funktionen Ergänzung der äquivalenten Funktionen bei den relationalen Operatoren
	Redaktionelle Anpassungen
Version 3.00	Fehlende Unter-Kapitel-Nummer 3.1 hinzugefügt.
	Neues Unterkapitel „Weitere Funktionen“ erfasst.
	Funktionen „filterResult“ und „length“ nach „Weitere Funktionen“ verschoben.
	Erweiterung der Funktion „filterResult“ um Werte-Filterung
	Konsolidierung von unaryMinus ⟶ wurde aus den arithmetischen Operatoren entfernt
	Erweiterung der Funktionen: abs, max, min, sum, errDiv
	Intervallarithmetik: Folgende Kapitel wurden erfasst:
	· Intervallarithmetik

- Arithmetische Operatoren – Intervallarithmetik
- Relationale Operatoren – Intervallarithmethik
- Mathematische Funktionen - Intervallarithmetik
- Logische-Funktionen – Intervallarithmetik
- Dimensions-Funktionen – Intervallarithmetik
- Aggregats-Funktionen – Intervallarithmetik
- Stammdaten-Funktionen – Intervallarithmetik
- String-Funktionen Intervallarithmetik
- Datentypprüfungen – Intervallarithmetik
- Konvertierungs-Funktionen – Intervallarithmetik
- Weitere Funktionen– Intervallarithmetik

Redaktionelle Anpassungen (bspw. Testdatensätze enden statt KI auf MO, Na wurde auf NA vereinheitlicht, ignoreNa Kapitel wurde inhaltlich verbessert)

Inhaltsverzeichnis

1 Allgemein

Die Oesterreichische Nationalbank überarbeitet ihre Meldeverarbeitungssysteme, dadurch kommt es zu Änderungen in der Prüfregelsyntax. Diese ist technologieunabhängig, auf sämtliche Standards anwendbar und einfach erweiterbar. Die neue Syntax ist somit vielseitig einsetzbar und genügt langfristig unterschiedlichsten Ansprüchen.

1.1 Grundlagen

Die Bezeichnung von Rechenregeln beginnt mit einem RR, darauf folgt eine 1- bis 25-stellige alphanumerische Kennung. Folgende Zeichen sind erlaubt A-Z, 0-9, _.

Beispiel:

RRBILANZSUMME

RR2357111317

RR29BSP0100510

RRBANK_01

Rechenregeln, welche im Formeleditor eingegeben werden, sind als mathematische, arithmetische Ausdrücke anzusehen. Aus diesem Grund können alle Bestandteile einer Rechenregel mit Grundrechnungsarten vereinigt werden. Logische Werte können mit den vordefinierten booleschen Standardfunktionen ausgewertet werden. Bei Bestandteilen, die Vektoren sind, werden die Grundrechnungsarten elementweise angewendet (siehe Kapitel 1.3).

Beispiel (Die einzelnen Bestandteile der Rechenregel - A, B, C, …):

*A + B / (C + D * E) * (((F + G) * H + I) * J + M)*

Bestandteile einer Rechenregel können mit Funktionen ausgewertet werden. Die Liste aller Funktionen ist im Anhang „Funktionen“ angeführt.

Beispiel:

math. Funktion: exp(A)

log. Funktion: and(A, B)

Datumsfunktion: getDay(A)

Die Parameter einer Funktion sind in Klammern anzugeben und durch Beistriche zu trennen. Jeder Parameter kann ein mathematischer Ausdruck (analog einer eigenen Rechenregel), eine Zeichenkette (unter Hochkomma = Charakter-String) oder ein Vektor sein.

1.2 Intervallarithmetik

Berechnung nach Intervallarithmetik bedeutet, dass die Werte mit einer Genauigkeit (@decimals) versehen werden, die ein Intervall um den Wert ("margin of error") bestimmt. Somit wird der Wert x durch das Intervall [x-0,5*10^(-@decimals) ; x+0,5*10^(-@decimals)] repräsentiert, wobei @decimals eine ganze Zahl ist, die die Genauigkeit repräsentiert.
Bsp.: @decimals=-3 entspricht einer Rundung auf 1000er und somit einer Genauigkeit von +/- 500.

Gleichheit ist definiert, als das Vorhandensein einer nicht leeren Schnittmenge der beiden Intervalle der Gleichung.

Für die Intervall-Arithmetik werden folgende @decimals-Werte (Genauigkeiten) angewandt:

ISIS-Konzepttyp	@decimals
Anzahl	0
Anzahl mit 4 Nachkommastellen	-3
Wert	-3
Prozent 2-stellig	2
Prozent 3-stellig	3
Prozent 4-stellig	4
Prozent 5-stellig	5
Prozent mind. 4-stellig	max(4, tatsächliche Anzahl an Nachkommastellen)
Prozent mind. 5-stellig	max(5, tatsächliche Anzahl an Nachkommastellen)

Die Funktionsweise der einzelnen Operatoren und Funktionen werden in eigenen Kapiteln beschrieben und sind an der Kapitel-Endung „[Funktionsname] **- Intervallarithmetik**“ erkennbar.

1.2.1 ignoreInterval

Das Attribut "ignoreInterval" schaltet von der Intervallarithmetik auf die Punkt-Arithmetik um. Für den Fall, dass Funktionen mit dem Parameter "ignoreInterval" aufgerufen werden, sollen sie äquivalent zu ihren Nicht-Intervallarithmetik-Pendants funktionieren.

Beispiele

Ausgangskonzept:

<IS01>= 1000, @decimals = -3

Aufruf	Ergebnis
[a; b]	[500; 1500]

<IS02>= 5000, @decimals = -3

Aufruf	Ergebnis
[c; d]	[4500; 5500]

k = 5

Aufruf	Ergebnis
[e; f]	[5; 5]

n = 3

Aufruf	Ergebnis
[g; h]	[3; 3]

Beispiel 1

Aufruf	Ergebnis
<IS01> + <IS02> bzw. plus(<IS01>,<IS02>)	[5000; 7000]
plus(<IS01>,<IS02>,"ignoreInterval")	6000
<IS01> - <IS02> bzw. minus(<IS01>,<IS02>)	[-5000; -3000]
minus(<IS01>,<IS02>,"ignoreInterval")	-4000
<IS01> * <IS02> bzw. mult(<IS01>,<IS02>)	[2250000; 8250000]

mult(<IS01>,<IS02>,"ignoreInterval")	5000000
<IS01> / <IS02> bzw. div(<IS01>,<IS02>)	[0,090909091; 0,333333333]
div(<IS01>,<IS02>,"ignoreInterval")	0,2

1.3 Logische Werte/Exceptional Values

Die logischen Werte TRUE und FALSE können in den Rechenregeln verwendet werden und als Ergebnis von Prüfregeln ausgegeben werden.

Es gibt zusätzlich in der Rechenregelsyntax die Ausprägungen NA (Not Available) und NaN (Not a Number) mit folgender Logik-Tabelle.

Rechnung	Ergebnis
NA + NA	NA
NaN + NaN	NaN
NA +NaN	NA
0/0	NaN
1/0	NaN
1 + NaN	NaN
1 + NA	NA

Die Logik-Tabelle lässt sich auf alle verketteten Funktionen erweitern. Das bedeutet, dass falls bei einer Berechnung ein NA oder NaN in den Werten vorkommt, ist das Ergebnis ein NA oder NaN.

1.3.1 ignoreNa

Um dieses Standardverhalten für NA zu ändern, kann pro Rechenregel global für alle enthaltenen Rechenfunktionen „ignoreNa“ gesetzt werden oder in der jeweiligen Rechenfunktion direkt der Ausdruck „ignoreNa“ angegeben werden. Dadurch werden in der Berechnung NAs ignoriert. Kommen in der Berechnung nur NAs vor, dann werden bei dieser Einstellung NAs durch eine numerische 0 ersetzt. Außer bei Multiplikationen und bei der Division beim Divisor - in diesem Fall wird ein NA mit dem neutralen Element (1) ersetzt, bei String-Funktionen (Kapitel 3.12 und *asChar*) wird NA mit “” (Leer-String) ersetzt.

Bei den Funktionen *and*, *or*, *xor* und *xnor* bewirkt das Setzen von ignoreNa, dass NAs immer durch eine numerische 0 ersetzt werden. 0 wird, wie in Kapitel Relationale Operatoren beschrieben, als *False* interpretiert.

Bei den Funktionen *aggExc, aggInc, sum, max* und *min* werden, wenn ignoreNa gesetzt ist, NAs ignoriert, außer wenn ausschließlich NAs in der Berechnung vorkommen, dann wird als Ergebnis ein NA geliefert.

Beispiele:

Ausgangskonzepte

<IS01>

MO	Wert
1200MO	12
3400MO	2
5600MO	NA

<IS02>

MO	Wert
1200MO	8
3400MO	NA
5600MO	NA

Beispiel 1:

Aufruf

sum(<IS01>, <IS02>, "ignoreNa“)

Ergebnis

MO	Wert
1200MO	20
3400MO	2
5600MO	NA

Beispiel 2:

Aufruf

max(<IS01>, <IS02>, "ignoreNa“)

Ergebnis

MO	Wert
1200MO	12
3400MO	2
5600MO	NA

Beispiel 3:

Aufruf

min(<IS01>, <IS02>, "ignoreNa“)

Ergebnis

MO	Wert
1200MO	8
3400MO	2
5600MO	NA

1.4 Arithmetische Operatoren

Als Operatorpräzedenz gelten die Standard-Rechenoperationen der Mathematik (in absteigender Priorität) in folgender Rangfolge:

1. Potenzierung
2. Multiplikation und Division („Punktrechnung“)
3. Addition und Subtraktion („Strichrechnung“)

Für jeden arithmetischen Operator gibt es eine Funktion, die äquivalent verwendet werden kann:

Operator	äquivalente Funktion	Beschreibung
+	plus()	führt eine Addition durch
-	minus()	führt eine Subtraktion durch
*	mult()	führt eine Multiplikation durch
/	div()	führt eine Division durch

^	pow()	führt eine Potenzierung durch

1.5 Arithmetische Operatoren – Intervallarithmetik

Folgende arithmetische Operatoren werden in der Intervallarithmetik verwendet:

Operator	äquivalente Funktion	Beschreibung
+	plus()	führt eine Addition durch
-	minus()	führt eine Subtraktion durch
*	mult()	führt eine Multiplikation durch
/	div()	führt eine Division durch

Die Funktionsweise der einzelnen Operatoren wurde in den folgenden Sub-Kapiteln in Berechnungen mit Skalaren, Berechnungen mit Vektoren und Berechnungen mit Skalaren mit Vektoren unterteilt.

1.5.1 Skalare – Intervallarithmetik

In der Intervallarithmetik können die arithmetischen Operatoren auf 2 bis n Skalare angewendet werden. Dabei soll das Intervall gemäß dem übergebenem Konzepttyp das Toleranzintervall berücksichtigt werden.

Funktionsaufruf

- Param1 + Param2
- plus(Param1, Param2)
- Param1 - Param2
- minus(Param1, Param2)
- Param1 * Param2
- mult(Param1, Param2)
- Param1 / Param2
- div(Param1, Param2)

Beispiel:

- <IS01> + <IS02>
- plus(<IS01>,<IS02>,"ignoreInterval")
- <IS01> - <IS02>
- minus(<IS01>,<IS02>,"ignoreInterval")
- <IS01> * <IS02>
- mult(<IS01>,<IS02>,"ignoreInterval")
- <IS01> / <IS02>
- div(<IS01>,<IS02>,"ignoreInterval")

Parameterbeschreibung

- Param1, Param2 sind skalare Konzepte oder ein Skalar

Funktionsbeschreibung

Fall 1: Berechnung mit 2 Konzepten

Sei <IS01> mit der Toleranz @decimals geflaggt, dann ist das Intervall [a;b] = [<IS01>-0,5*10^(-@decimals);<IS01>+0,5*10^(-@decimals)].

Sei <IS02> mit der Toleranz @decimals geflaggt, dann ist das Intervall [c;d] = [<IS02>-0,5*10^(-@decimals);<IS02>+0,5*10^(-@decimals)].

[a;b] + [c;d] = [a+c; b+d]
[a;b] - [c;d] = [a-d, b-c]
[a;b] * [c;d] = [min (a*c, a*d, b*c, b*d); max (a*c, a*d, b*c, b*d)]
[a;b] / [c;d] = [min (a/c, a/d, b/c, b/d); max (a/c, a/d, b/c, b/d)]

Die Division durch ein Intervall, die 0 enthält, ist in der Intervallarithmetik nicht definiert.

Fall 2: Berechnung mit 1 Konzept und 1 Skalar

Sei <IS01> mit @decimals1 geflaggt, dann ist das Intervall [a;b] = [<IS01>-0,5*10^(-@decimals1);<IS01>+0,5*10^(-@decimals1)].

Sei k ein Skalar, dann ist @decimals2=INF (unendlich) und damit [c;d] = [k;k]

Berechnungen siehe oben

Fall 3: Berechnung mit 2 Skalaren

Sei x ein Skalar, dann ist @decimals1=INF und damit [a;b] = [x;x]

Sei k ein Skalar, dann ist @decimals2=INF und damit [c;d] = [k;k]

Berechnung siehe oben und soll der "Intervallarithmetik-losen" Berechnung entsprechen.

Fall 4: Berechnung mit plus(), minus(), mult(), div()

Fall 4a: plus(), minus(), mult(), div() - Aufruf ohne Parameter

Für den Fall, dass die Funktionen ohne Parameter aufgerufen werden, sollen sie wie in den Fällen 1-3 äquivalent ihrem binären Operator funktionieren.

Fall 4b: plus(), minus(), mult(), div() - Aufruf mit Parameter "ignoreInterval"

Für den Fall, dass die Funktionen mit dem Parameter "ignoreInterval" aufgerufen werden, sollen sie äquivalent zu ihren Nicht-Intervallarithmetik-Pendants funktionieren.

Beispiele

Ausgangskonzepte:

<IS01>= 1000, @decimals = -3 ⟶ Intervall: [500; 1500]

<IS02>= 5000, @decimals = -3 ⟶ Intervall: [4500; 5500]

k = 5 ⟶ Intervall: [5;5]

x = 3 ⟶ Intervall: [3;3]

Beispiel 1

Aufruf	Ergebnis
<IS01> + <IS02> bzw. plus(<IS01>,<IS02>)	[5000; 7000]
plus(<IS01>,<IS02>,"ignoreInterval")	6000
<IS01> - <IS02> bzw. minus(<IS01>,<IS02>)	[-5000; -3000]
minus(<IS01>,<IS02>,"ignoreInterval")	-4000
<IS01> * <IS02> bzw. mult(<IS01>,<IS02>)	[2250000; 8250000]
mult(<IS01>,<IS02>,"ignoreInterval")	5000000
<IS01> / <IS02> bzw. div(<IS01>,<IS02>)	[0,090909091; 0,333333333]
div(<IS01>,<IS02>,"ignoreInterval")	0,2

Beispiel 2

Aufruf	Ergebnis
<IS02> + k bzw. plus(<IS02>,k)	[4505; 5505]
plus(<IS02>,k,"ignoreInterval")	5005
<IS02> - k bzw. minus(<IS02>,k)	[4495; 5495]
minus(<IS02>,k,"ignoreInterval")	4995
<IS02> * k bzw. mult(<IS02>,k)	[22500; 27500]
mult(<IS02>,k,"ignoreInterval")	25000
<IS02> / k bzw. div(<IS02>,k)	[900; 1100]
div(<IS02>,k,"ignoreInterval")	1000

Beispiel 3

Aufruf	Ergebnis
k + x bzw. plus(k,x)	[8;8]
plus(k,x,"ignoreInterval")	8
minus(k,x,"ignoreInterval")	[2;2]
minus(k,x,"ignoreInterval")	2
k * x bzw. mult(k,x)	[15; 15]
mult(k,x,"ignoreInterval")	15
k / x bzw. div(k,x)	[1,6666666; 1,6666666]
div(k,x,"ignoreInterval")	1,6666666

1.5.2 Vektoren – Intervallarithmetik

Funktionsbeschreibung

Pro Zeile des Vektors soll eine skalare Operation der Grundrechnungsart ausgeführt werden. Dabei soll für jede Zeile das mitgegebene Toleranzintervall verwendet werden.

Funktionsaufruf

- Param1 + Param2
- plus(Param1, Param2)
- Param1 - Param2
- minus(Param1, Param2)
- Param1 * Param2
- mult(Param1, Param2)
- Param1 / Param2
- div(Param1, Param2)

Beispiel:

- <IS01> + <IS02>
- plus(<IS01>,<IS02>,"ignoreInterval")
- <IS01> - <IS02>
- minus(<IS01>,<IS02>,"ignoreInterval")
- <IS01> * <IS02>
- mult(<IS01>,<IS02>,"ignoreInterval")
- <IS01> / <IS02>

- div(<IS01>,<IS02>,"ignoreInterval")

Parameterbeschreibung

- Param1, Param2 sind Vektoren

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

<IS02>, @decimals=-3

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	10000
5600MO	AT	1000

Beispiel 1

Aufruf:

<IS01> + <IS02>
bzw.
plus(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	[5000; 7000]
3400MO	DE	[8000; 10000]
5600MO	DE	[14000; 16000]

5600MO	AT	[2000; 4000]

Beispiel 2

Aufruf:

plus(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	6000
3400MO	DE	9000
5600MO	DE	15000
5600MO	AT	3000

Beispiel 3

Aufruf:

<IS01> - <IS02>
bzw.
minus(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	[-5000; -3000]
3400MO	DE	[2000; 4000]
5600MO	DE	[-6000; -4000]
5600MO	AT	[0; 2000]

Beispiel 4

Aufruf:

minus(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	-4000

3400MO	DE	3000
5600MO	DE	-5000
5600MO	AT	1000

Beispiel 5

Aufruf:

<IS01> * <IS02>
bzw.
mult(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	[2250000; 8250000]
3400MO	DE	[13750000; 22750000]
5600MO	DE	[42750000; 57750000]
5600MO	AT	[750000; 3750000]

Beispiel 6

Aufruf:

mult(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	5000000
3400MO	DE	18000000
5600MO	DE	50000000
5600MO	AT	2000000

Beispiel 7

Aufruf:

<IS01> / <IS02>
bzw.
div(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	[0,090909091; 0,333333333]
3400MO	DE	[1,571428571; 2,6]
5600MO	DE	[0,428571429; 0,578947368]
5600MO	AT	[1; 5]

Beispiel 8

Aufruf:

div(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	0,2
3400MO	DE	2
5600MO	DE	0,5
5600MO	AT	2

1.5.3 Skalare mit Vektoren – Intervallarithmetik

Funktionsbeschreibung

Der Skalar soll auf die Dimensionalität des Vektors mit n (=Anzahl der Dimensionen) gebracht werden, indem alle n Dimensionen mit dem Intervall des Skalars aufgefüllt werden. Danach soll pro Zeile der Vektoren eine skalare Operation der Grundrechnungsart ausgeführt werden. Dabei soll für jede Zeile das mitgegebene Toleranzintervall verwendet werden.

Funktionsaufruf

- Param1 + Param2
- plus(Param1, Param2)
- Param1 - Param2

- minus(Param1, Param2)
- Param1 * Param2
- mult(Param1, Param2)
- Param1 / Param2
- div(Param1, Param2)

Beispiel:

- <IS01> + <IS02>
- plus(<IS01>,<IS02>,"ignoreInterval")
- <IS01> - <IS02>
- minus(<IS01>,<IS02>,"ignoreInterval")
- <IS01> * <IS02>
- mult(<IS01>,<IS02>,"ignoreInterval")
- <IS01> / <IS02>
- div(<IS01>,<IS02>,"ignoreInterval")

Parameterbeschreibung

- Param1, Param2 sind skalare Konzepte oder ein Skalar

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

k = 10

Beispiel 1

Aufruf:

<IS01> + k
bzw.
plus(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	[510; 1510]
3400MO	DE	[5510; 6510]
5600MO	DE	[4510; 5510]
5600MO	AT	[1510; 2510]

Beispiel 2

Aufruf:

plus(<IS01>,k,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	1010
3400MO	DE	6010
5600MO	DE	5010
5600MO	AT	2010

Beispiel 3

Aufruf:

<IS01> - k
bzw.
minus(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	[490; 1490]

3400MO	DE	[5490; 6490]
5600MO	DE	[4490; 5490]
5600MO	AT	[1490; 2490]

Beispiel 4

Aufruf:

minus(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	990
3400MO	DE	5990
5600MO	DE	4990
5600MO	AT	1990

Beispiel 5

Aufruf:

<IS01> * k
bzw.
mult(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	[5000; 15000]
3400MO	DE	[55000; 65000]
5600MO	DE	[45000; 55000]
5600MO	AT	[15000; 25000]

Beispiel 6

Aufruf:

mult(<IS01>,k,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	10000
3400MO	DE	60000
5600MO	DE	50000
5600MO	AT	20000

Beispiel 7

Aufruf:

<IS01> / k
bzw.
div(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	[50; 150]
3400MO	DE	[550; 650]
5600MO	DE	[450; 550]
5600MO	AT	[150; 250]

Beispiel 8

Aufruf:

div(<IS01>,k,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	100
3400MO	DE	600

5600MO	DE	500
5600MO	AT	200

1.6 Relationale Operatoren

Für jeden relationalen Operator gibt es eine Funktion, die äquivalent verwendet werden kann. Gültige relationale Operatoren in dieser Syntax sind:

Operator	äquivalente Funktion	Beschreibung
==	eq	Überprüft Gleichheit
!=	neq	Überprüft Ungleichheit
!	unaryNot	Nicht
&&	and	Logisches UND
||	or	Logisches ODER
<	lt	Kleiner
<=	le	Kleiner gleich
>	gt	Größer
>=	ge	Größer gleich

Grundsätzlich ist die Operatorpräzedenz hier als linksassoziativ anzunehmen.

Zusätzlich wurde ein Castprozess in den folgenden speziellen Operatoren implementiert. Für die Operatoren <, <=, >, >=, == und != gibt es eine einheitliche Implementierung. Wenn ein String mit einer Zahl verglichen wird, versucht die Engine den String in eine Zahl zu casten. Gelingt das nicht, kommt ein Rechenfehler!

Beifolgenden logischen Operatoren AND, OR, !, XOR, und XNOR wird auch ein cast durchgeführt, falls ein Boolean mit einer Zahl oder eine Zahl mit einer Zahl verkettet wird. Hierfür werden die Zahl "0" als "FALSE" interpretiert jede andere Zahl wird als "TRUE" interpretiert (auch beispielsweise bei 0,00000000001).

1.7 Relationale Operatoren – Intervallarithmethik

Für jeden relationalen Operator gibt es eine Funktion, die äquivalent verwendet werden kann. Gültige relationale Operatoren in der Intervallarithmetik-Syntax sind:

Operator	äquivalente Funktion	Beschreibung
==	eq	Überprüft Gleichheit
!=	neq	Überprüft Ungleichheit
!	unaryNot	Nicht
&&	and	Logisches UND
||	or	Logisches ODER
<	lt	Kleiner
<=	le	Kleiner gleich
>	gt	Größer
>=	ge	Größer gleich

In den folgenden Unterkapiteln werden die Operatoren genauer beschrieben.

1.7.1 Skalare – Gleichheit – Intervallarithmetik

Funktionsbeschreibung

Fall 1: Beiden Argumenten wird ein Intervall mitgegeben

Sei <IS01> mit @decimals1 geflaggt, dann ist das Intervall [a;b] = [<IS01>-0,5*10^(-@decimals1);<IS01>+0,5*10^(-@decimals1)].

Sei <IS02> mit @decimals2 geflaggt, dann ist das Intervall [c;d] = [<IS02>-0,5*10^(-@decimals2);<IS02>+0,5*10^(-@decimals2)].

<IS01> == <IS02> genau dann, wenn [a;b] und [c;d] **keine** leere Schnittmenge haben.

Fall 2: Bei einem der beiden Argumente wird ein Intervall mitgegeben

Vergleich Konzept mit Zahl → hier soll wie in Skalare – Intervallarithmetik dem Skalar ein "Nullintervall" mitgegeben werden (Sei k eine Zahl, dann ist @decimals=INF und damit [k;k] das Intervall). → Fall 1

Funktionsaufruf

- Param1 == Param2
- eq(Param1, Param2)
- Param1 != Param2
- neq(Param1, Param2)

Beispiel:

- <IS01> == <IS02>
- eq(<IS01>,<IS02>,"ignoreInterval")
- <IS01> !=<IS02>
- neq(<IS01>,<IS02>,"ignoreInterval")

Parameterbeschreibung

- Param1, Param2 sind skalare Konzepte oder ein Skalar

Beispiele

Ausgangskonzept:

<IS01> = 1000, @decimals = -3 ⟶ Intervall: [500; 1500]

<IS02>= 5000, @decimals = -3 ⟶ Intervall: [4500; 5500]

<IS03> = 1500; @decimals= 4 ⟶ Intervall: [1499,99995; 1500,00005]

k = 5 ⟶ Intervall: [5;5]

x = 3 ⟶ Intervall: [3;3]

Beispiel 1

Aufruf	Ergebnis
<IS01> == <IS02> bzw. eq(<IS01>,<IS02>)	False
eq(<IS01>,<IS02>,"ignoreInterval")	False
<IS01> != <IS02> bzw. neq(<IS01>,<IS02>)	True
neq(<IS01>,<IS02>,"ignoreInterval")	True
<IS01> == <IS3> bzw. eq(<IS01>,<IS3>)	True
eq(<IS01>,<IS3>,"ignoreInterval")	False
<IS01> != <IS3> bzw. neq(<IS01>,<IS3>)	False
neq(<IS01>,<IS3>,"ignoreInterval")	True

<IS02> == <IS3> bzw. eq(<IS02>,<IS3>)	False
eq(<IS02>,<IS3>,"ignoreInterval")	False
<IS02> != <IS3> bzw. neq(<IS02>,<IS3>)	True
neq(<IS02>,<IS3>,"ignoreInterval")	True

Beispiel 2

Aufruf	Ergebnis
k == x bzw. eq(k,x)	False
eq(k,x,"ignoreInterval")	False
k != n bzw. neq(k,x)	True
neq(k,x,"ignoreInterval")	True
<IS01> == k bzw. eq(<IS01>,k)	False
eq(<IS01>,k,"ignoreInterval")	False
<IS01> != k bzw. neq(<IS01>,k)	True
neq(<IS01>,k,"ignoreInterval")	True

1.7.2 Vektoren – Gleichheit – Intervallarithmetik

Funktionsbeschreibung

Pro Zeile des Vektors soll eine skalare Operation der Gleichheitsprüfung ausgeführt werden. Dabei soll für jede Zeile das mitgegebene Toleranzintervall verwendet werden.

Funktionsaufruf

- Param1 == Param2
- eq(Param1, Param2)
- Param1 != Param2
- neq(Param1, Param2)

Beispiel:

- <IS01> == <IS02>
- eq(<IS01>,<IS02>,"ignoreInterval")
- <IS01> !=<IS02>
- neq(<IS01>,<IS02>,"ignoreInterval")

Parameterbeschreibung

- Param1, Param2 sind Vektoren

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

<IS02>, @decimals=-3

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	4700
5600MO	AT	1000

Beispiel 1

Aufruf:

<IS01> == <IS02>
bzw.
eq(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	False
5600MO	DE	True

5600MO	AT	True

Beispiel 2

Aufruf:

eq(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	False
5600MO	DE	False
5600MO	AT	False

Beispiel 3

Aufruf:

<IS01> != <IS02>
bzw.
neq(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	True
5600MO	DE	False
5600MO	AT	False

Beispiel 4

Aufruf:

neq(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	True

3400MO	DE	True
5600MO	DE	True
5600MO	AT	True

1.7.3 Skalare mit Vektoren – Gleichheit – Intervallarithmetik

Funktionsbeschreibung

Der Skalar soll auf die Dimensionalität des Vektors mit n (=Anzahl der Dimensionen) gebracht werden, indem alle n Dimensionen mit dem Intervall des Skalars aufgefüllt werden. Danach soll pro Zeile der Vektoren soll eine skalare Operation der Gleichheit ausgeführt werden. Dabei soll für jede Zeile das mitgegebene Toleranzintervall verwendet werden.

Funktionsaufruf

- Param1 == Param2
- eq(Param1, Param2)
- Param1 != Param2
- neq(Param1, Param2)

Beispiel:

- <IS01> == <IS02>
- eq(<IS01>,<IS02>,"ignoreInterval")
- <IS01> !=<IS02>
- neq(<IS01>,<IS02>,"ignoreInterval")

Parameterbeschreibung

- Param1, Param2 sind Vektoren oder Skalare

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

k = 1010

Beispiel 1

Aufruf:

<IS01> == k
bzw.
eq(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	False
5600MO	DE	False
5600MO	AT	False

Beispiel 2

Aufruf:

eq(<IS01>,k,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	False
5600MO	DE	False
5600MO	AT	False

Beispiel 3

Aufruf:

<IS01> != k
bzw.
neq(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	True
5600MO	DE	True
5600MO	AT	True

Beispiel 4

Aufruf:

neq(<IS01>,k,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	True
5600MO	DE	True
5600MO	AT	True

1.7.4 Skalare – Ungleichheit – Intervallarithmetik

Funktionsbeschreibung

Fall 1: Beiden Argumenten wird ein Intervall mitgegeben

Sei <IS01> mit @decimals1 geflaggt, dann ist das Intervall [a;b] = [<IS01>-0,5*10^(-@decimals1);<IS01>+0,5*10^(-@decimals1)].

Sei <IS02> mit @decimals2 geflaggt, dann ist das Intervall [c;d] = [<IS02>-0,5*10^(-@decimals2);<IS02>+0,5*10^(-@decimals2)].

- <IS01> < <IS02> genau dann, wenn es einen Wert in [a;b] gibt, der **kleiner** ist als ein Wert in [c;d].
- <IS01> <= <IS02> genau dann <IS01> < <IS02> (s.o.) oder <IS01> == <IS02> (i.e [a;b] und [c;d] **keine** leere Schnittmenge haben)
- <IS01> > <IS02> genau dann, wenn es einen Wert in [a,b] gibt, der **größer** ist als ein Wert in [c;d].
- <IS01> >= <IS02> genau dann <IS01> > <IS02> (s.o.) oder <IS01> == <IS02> (i.e [a;b] und [c;d] **keine** leere Schnittmenge haben)

Fall 2: Bei einem der beiden Argumente wird ein Intervall mitgegeben

Vergleich Konzept mit Zahl → hier soll wie in Skalare – Intervallarithmetik dem Skalar ein "Nullintervall" mitgegeben werden (Sei k eine Zahl, dann ist @decimals=INF und damit [k;k] das Intervall). → Fall 1

Funktionsaufruf

- Param1 > Param2
- gt(Param1,Param2,Param1)
- Param1 >= Param2
- ge(Param1, Param2)
- Param1 < Param2
- lt(Param1,Param2,Param1)
- Param1 <= Param2
- le(Param1, Param2)

Beispiel:

- <IS01> > <IS02>
- gt(<IS01>,<IS02>,"ignoreInterval")
- <IS01> >= <IS02>
- ge(<IS01>,<IS02>,"ignoreInterval")
- <IS01> < <IS02>
- lt(<IS01>,<IS02>,"ignoreInterval")
- <IS01> <= <IS02>
- le(<IS01>,<IS02>,"ignoreInterval")

Parameterbeschreibung

- Param1, Param2 sind skalare Konzepte oder ein Skalar

Beispiele

Ausgangskonzept:

<IS01>= 1000, @decimals = -3 → Intervall: [500; 1500]

<IS02>= 5000, @decimals = -3 → Intervall: [4500; 5500]

<IS03> = 1500; @decimals= 4 → Intervall: [1499,99995; 1500,00005]

k = 500 → Intervall: [500;500]

x = 300 → Intervall: [300;300]

Beispiel 1

Aufruf	Ergebnis
<IS01> > <IS02> bzw. gt(<IS01>,<IS02>)	False
gt(<IS01>,<IS02>,"ignoreInterval")	False
<IS01> >= <IS02> bzw. ge(<IS01>,<IS02>)	False
ge(<IS01>,<IS02>,"ignoreInterval")	False
<IS01> > <IS3> bzw. gt(<IS01>,<IS3>)	True
gt(<IS01>,<IS3>,"ignoreInterval")	False
<IS01> >= <IS3> bzw. ge(<IS01>,<IS3>)	True
ge(<IS01>,<IS3>,"ignoreInterval")	False
<IS02> > <IS3> bzw. gt(<IS02>,<IS3>)	True
gt(<IS02>,<IS3>,"ignoreInterval")	True
<IS02> >= <IS3> bzw. ge(<IS02>,<IS3>)	True
ge(<IS02>,<IS3>,"ignoreInterval")	True
<IS01> < <IS02> bzw. lt(<IS01>,<IS02>)	True
lt(<IS01>,<IS02>,"ignoreInterval")	True
<IS01> <= <IS02> bzw. le(<IS01>,<IS02>)	True
le(<IS01>,<IS02>,"ignoreInterval")	True
<IS01> < <IS3> bzw. lt(<IS01>,<IS3>)	True
le(<IS01>,<IS3>,"ignoreInterval")	True
<IS01> <= <IS3> bzw. le(<IS01>,<IS3>)	True
le(<IS01>,<IS3>,"ignoreInterval")	True
<IS02> < <IS3> bzw. lt(<IS02>,<IS3>)	False
lt(<IS02>,<IS3>,"ignoreInterval")	False
<IS02> <= <IS3> bzw. le(<IS02>,<IS3>)	False

le(<IS02>,<IS3>,"ignoreInterval")	False

Beispiel 2

Aufruf	Ergebnis
k > x bzw. gt(k,x)	True
gt(k,x,"ignoreInterval")	True
k >= x bzw. ge(k,x)	True
ge(k,x,"ignoreInterval")	True
k < x bzw. lt(k,x)	False
lt(k,x,"ignoreInterval")	False
k <= x bzw. le(k,x)	False
le(k,x,"ignoreInterval")	False

Beispiel 3

Aufruf	Ergebnis
<IS01> > k bzw. gt(<IS01>,k)	True
gt(<IS01>,k,"ignoreInterval")	True
<IS01> >= k bzw. ge(<IS01>,k)	True
ge(<IS01>,k,"ignoreInterval")	True
<IS01> < k bzw. lt(<IS01>,k)	False
lt(<IS01>,k,"ignoreInterval")	False
<IS01> <= k bzw. le(<IS01>,k)	True
le(<IS01>,k,"ignoreInterval")	False

1.7.5 Vektoren – Ungleichheit – Intervallarithmetik

Funktionsbeschreibung

Pro Zeile des Vektors soll eine skalare Operation der Ungleichheitsprüfung ausgeführt werden. Dabei soll für jede Zeile das mitgegebene Toleranzintervall verwendet werden.

Funktionsaufruf

- Param1 > Param2
- gt(Param1,Param2,Param1)
- Param1 >= Param2
- ge(Param1, Param2)
- Param1 < Param2
- lt(Param1,Param2,Param1)
- Param1 <= Param2
- le(Param1, Param2)

Beispiel:

- <IS01> > <IS02>
- gt(<IS01>,<IS02>,"ignoreInterval")
- <IS01> >= <IS02>
- ge(<IS01>,<IS02>,"ignoreInterval")
- <IS01> < <IS02>
- lt(<IS01>,<IS02>,"ignoreInterval")
- <IS01> <= <IS02>
- le(<IS01>,<IS02>,"ignoreInterval")

Parameterbeschreibung

- Param1, Param2 sind Vektoren

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	1500

<IS02>, @decimals=-3

MO	LD	Werte

1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	4700
5600MO	AT	1000

Beispiel 1

Aufruf:

<IS01> ><IS02>
bzw.
gt(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	True
5600MO	DE	True
5600MO	AT	True

Beispiel 2

Aufruf:

gt(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	True
5600MO	DE	True
5600MO	AT	True

Beispiel 3

Aufruf:

<IS01> >= <IS02>
bzw.
ge(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	True
5600MO	DE	True
5600MO	AT	True

Beispiel 4

Aufruf:

ge(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	True
5600MO	DE	True
5600MO	AT	True

Beispiel 5

Aufruf:

<IS01> < <IS02>
bzw.
lt(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	False
5600MO	DE	True

5600MO	AT	True

Beispiel 6

Aufruf:

lt(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	False
5600MO	DE	False
5600MO	AT	False

Beispiel 7

Aufruf:

<IS01> <= <IS02>
bzw.
le(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	False
5600MO	DE	True
5600MO	AT	True

Beispiel 8

Aufruf:

le(<IS01>,<IS02>,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	True

3400MO	DE	False
5600MO	DE	False
5600MO	AT	False

1.7.6 Skalare mit Vektoren – Ungleichheit – Intervallarithmetik

Funktionsbeschreibung

Der Skalar soll auf die Dimensionalität des Vektors mit n (=Anzahl der Dimensionen) gebracht werden, indem alle n Dimensionen mit dem Intervall des Skalars aufgefüllt werden. Danach soll pro Zeile der Vektoren soll eine skalare Operation der Ungleichheit ausgeführt werden. Dabei soll für jede Zeile das mitgegebene Toleranzintervall verwendet werden.

Funktionsaufruf

- Param1 > Param2
- gt(Param1,Param2,Param1)
- Param1 >= Param2
- ge(Param1, Param2)
- Param1 < Param2
- lt(Param1,Param2,Param1)
- Param1 <= Param2
- le(Param1, Param2)

Beispiel:

- <IS01> > <IS02>
- gt(<IS01>,<IS02>,"ignoreInterval")
- <IS01> >= <IS02>
- ge(<IS01>,<IS02>,"ignoreInterval")
- <IS01> < <IS02>
- lt(<IS01>,<IS02>,"ignoreInterval")
- <IS01> <= <IS02>
- le(<IS01>,<IS02>,"ignoreInterval")

Parameterbeschreibung

- Param1, Param2 sind Vektoren oder Skalare

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	1510
5600MO	DE	500
5600MO	AT	2000

k = 1010

Beispiel 1

Aufruf:

<IS01> > k
bzw.
gt(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	True
5600MO	DE	False
5600MO	AT	True

Beispiel 2

Aufruf:

gt(<IS01>,k,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	True
5600MO	DE	False

5600MO	AT	True

Beispiel 3

Aufruf:

<IS01> >= k
bzw.
ge(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	True
5600MO	DE	False
5600MO	AT	True

Beispiel 4

Aufruf:

ge(<IS01>,k,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	True
5600MO	DE	False
5600MO	AT	True

Beispiel 5

Aufruf:

<IS01> < k
bzw.
lt(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	False
5600MO	DE	True
5600MO	AT	False

Beispiel 6

Aufruf:

lt(<IS01>,k,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	False
5600MO	DE	True
5600MO	AT	False

Beispiel 7

Aufruf:

<IS01> <= k
bzw.
le(<IS01>,k)

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	True
5600MO	DE	True
5600MO	AT	False

Beispiel 8

Aufruf:

le(<IS01>,k,"ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	True
3400MO	DE	False
5600MO	DE	True
5600MO	AT	False

1.8 Definition des Rechenraums

Für die Berechnungen der Rechenregelsyntax wird ein Vektorraum (mathematische Definition) zu Grunde gelegt und somit gelten die daraus resultierenden Regeln (falls alle Bestandteile der RR die gleichen Dimensionen haben) für die Rechen-Engine:

1. Es gelte das *Assoziativgesetz* (bzgl. der Vektoraddition und –Multiplikation).
2. Es gelte das *Kommutativgesetz* (bzgl. der Vektoraddition und –Multiplikation).
3. Es gelte das *Distributivgesetz* (bzgl. der Vektoraddition und –Multiplikation).
4. Es existiere ein *neutrales Element* (bzgl. der Vektoraddition und –Multiplikation).
5. Es existiere ein *inverses Element* (bzgl. der Vektoraddition und –Multiplikation).

1.9 Rechnen mit Vektoren

Konzepte können mehrere Dimensionen haben, dadurch ist bei der Berechnung von Prüfregeln eine Vektorrechnung möglich.

Folgende Arithmetik ist umgesetzt.

1. $(1) + (2) = (3)$ // 1 + 2 = 3
2. $\begin{pmatrix}1\\1\\1\end{pmatrix} + (1) = \begin{pmatrix}2\\2\\2\end{pmatrix}$
3. (1)[AT] + (2)[DE] = (3)
4. $\begin{pmatrix}1\\1\\1\end{pmatrix}\begin{bmatrix}AT\\DE\\FR\end{bmatrix} + \begin{pmatrix}2\\3\end{pmatrix}\begin{bmatrix}AT\\DE\end{bmatrix} = \begin{pmatrix}3 & [AT])\\4 & [DE]\\NA & [FR])\end{pmatrix}$
5. $\begin{pmatrix}1\\NA\\1\end{pmatrix}\begin{bmatrix}AT\\DE\\FR\end{bmatrix} + \begin{pmatrix}2\\3\\4\end{pmatrix}\begin{bmatrix}AT\\DE\\FR\end{bmatrix} = \begin{pmatrix}3 & [AT]\\NA & [DE]\\5 & [FR]\end{pmatrix}$

1.10 Implementierung

Die Calculation-Engine ist in Java implementiert und besitzt Abhängigkeiten zu Apache-commons und zu slf4j Logging Frameworks.

Der Parserteil der Calculation-Engine wurde mit Hilfe von JavaCC (Version 7.0) erzeugt.

Für die Rechengenauigkeit wird die Java-Klasse BigDecimal verwendet. Der Vorteil gegenüber float und double ist, dass die Berechnung genauer ist. Dies liegt darin, dass die Vor- und Nachkomma-Stellen in einem BigDecimal-Objekt exakt abgelegt werden und nicht näherungsweise, wie es bei Fließkommadarstellungen.

Alle Werte die den Konzepttyp „Prozent“ haben, werden wie gemeldet in der Prüfung gerechnet, dies hängt von der Erhebung ab. Somit kann 100 % entweder 100 oder 1 sein.

Alle gemeldeten Werte mit einem anderen Konzepttyp werden in ihre echten Werte umgerechnet. Beispielhaft wird das Konzept IS893, Konzepttyp „Wert in Tausend“, mit einem Meldewert von 1 auf den echten Wert 1.000 umgerechnet und erst dann für die Berechnung von Prüfungen verwendet.

2 Bestandteile einer Rechenregel

Rechenregeln, welche im Formeleditor eingegeben werden, sind als mathematische, arithmetische Ausdrücke anzusehen. Sie können sich aus folgenden Bestandteilen zusammensetzen.

- Konzepten,
- Rechenregeln,
- Bestandteile (Konzepte, Rechenregeln) mit Dimensionen

2.1 Konzepte

Konzepte beginnen immer mit der 2-stelligen Bereichskennung und spezifizieren die Herkunft. IS steht für gemeldete Konzepte. Dahinter ist die datenspezifische Identifikation angeführt.

Beispiel:

IS1123581321

IS1247111622

2.2 Rechenregeln

Rechenregeln beginnen mit einem RR, darauf folgt eine 1- bis 25-stellige alphanumerische Kennung. Die Verwendung ist so zu verstehen, dass aufgerufene Rechenregeln berechnet und das Ergebnis weiterverwendet wird. Folgendes Beispiel beschreibt diesen Fall:

Die Rechenregel *RRFINUS_V4208H* besteht aus der Syntax:

1+ RRSUMME < ISFIN1235813

Weiters hat *RRSUMME* die nachstehende Syntax:

ISFIN0100012 + ISFIN0100014

Wird nun die Prüfregel *RRFINUS_V4208H* ausgeführt, wird zuerst die *RRSUMME* berechnet und das Ergebnis wird danach in *RRFINUS_V4208H* für die weitere Berechnung herangezogen.

Somit gilt pro Rechenregel mit referenzierten Rechenregeln, dass zuerst die referenzierten Rechenregeln berechnet werden, um das Rechenergebnis danach in die ursprüngliche Rechenregel zur weiteren Berechnung einzusetzen.

2.3 Bestandteile mit Dimensionen

Konzepte mit Dimensionsausprägungen sind in spitzen Klammern ("<", ">") zu schreiben, wobei die Dimensionen mit mindestens einem Leerzeichen (oder Tabulator) zu trennen und mit einem „=" zu identifizieren sind. Die Dimensionen und die Ausprägungen sind dem OeNB-Wiki zu entnehmen.

Beispiel:

<IS12481632 WG=EUR>

<RRQuodEratDemonstrandum LD=AT MP=20130101>

<IS12481632 SC_Geschaeftsfall_Waehrung_Code=EUR>

<RR29BSP0100510 SC_Sitzland_Code=AT >

Konzepte und Rechenregeln ohne Dimensionsausprägungen können auch in spitzen Klammern ("<", ">") dargestellt werden.

Beispiel:

<IS1123581321>

<IS1247111622>

<RRBILANZSUMME1>

Bestandteile mit Dimensionen können ohne weitere Verarbeitung in Funktionen aufgerufen werden.

Beispiel:

max(<RR11135917 LD=PL>)

2.3.1 SCS-Mengen

Um Mengen aus dem Mengenbildungssystem der OeNB (SCS - Set Construction System) explizit in Rechenregeln zu verwenden, ist an die jeweilige Dimension der Ausdruck „_G“ vor den Zuweisungsparameter anzuhängen.

Beispiel:

<IS1123581321 WG_G=WG_OHNE_EURO>

<IS1247111622 WA_G=WA_WPSC>

max(<RR11135917 LD_G=SEKTOR_EU>)

Die explizite Angabe aller Elemente einer Dimension ist mit dem Schlüsselwort „ALLE“ möglich und benötigt auch den Ausdruck „_G“. Der Ausdruck „_G=ALLE“ muss jedoch nicht zwingend angegeben werden, da es die Syntax unnötig vergrößert und vom System automatisch für nicht angegebene Dimensionen angenommen wird.

Beispiel:

max(<IS1123581321 LD_G=ALLE>)

max(<IS1123581321>)

beide Rechenregeln liefern das identische Ergebnis

2.3.2 Gruppierungen

Statt einer SCS-Menge ist es weiters möglich Einzelelemente einer Dimension mittels eckigen Klammern anzugeben.

Beispiel:

<IS1123581321 WG_G=[EUR, USD, GBP]>

<IS19871512 LD_G=[AT, DE, PL]>

Mit geschwungenen Klammern {,} können mehrere SCS-Sektoren eingegeben werden. Es wird die Vereinigungsmenge gebildet und jedes Element nur einmal selektiert

Beispiel:

<IS1123581321 WG_G={WG_OHNE_EURO, WG_MIT_EURO}>

<IS1247111622 WA_G={WA_WPSC, WA_KRSC}>

max(<RR11135917 LD_G={SEKTOR_EU, SEKTOR_NONEU}>)

2.3.3 Dimensionsvergleich

Durch die Eingabe eines "_D" kann nach der Eingabe des "=" eine Dimension des gleichen Konzeptes eingegeben werden. Diese Auswahl ist zur Filterung der Dimensionen von Konzepten angedacht. Weiters kann bei "_D" auch ein „!=" benutzt werden.

- *<IS01123 LD_D=LDC>* - Es werden alle Datensätze des Konzeptes IS01123 zur Berechnung verwendet, bei denen die Dimensionsausprägung Land (LD) und Land-Counterpart (LDC) *gleich* sind.
- *<IS01123 LD_D!=LDC>* - Es werden alle Datensätze des Konzeptes IS01123 zur Berechnung verwendet, bei denen die Dimensionsausprägung Land (LD) und Land-Counterpart (LDC) *nicht gleich* sind.

2.3.4 Dimension Meldeperiode (MP)

Bei Bestandteilen mit Dimensionen ist die Dimension „MP" – Meldeperiode gesondert zu betrachten. Im Regelfall werden über die Rechenregelsyntax nur Positionen innerhalb einer Meldung geprüft und damit Werte innerhalb einer Meldeperiode.

Beispiel:

IS153200001 < IS153300002

Fixe Meldeperiodenangabe

Es lassen sich jedoch auch explizit Meldeperioden mit dem Datumsformat „JJJJMMTT" angeben

Beispiel:

<IS153200001 MP=20131231> < IS153300001

Beim oberhalb beschriebenen Beispiel wird verglichen, ob der Wert der Meldung pro Periode kleiner als der Wert aus der Meldeperiode Dezember 2013 (Stichtag 31.12.2013) ist.

Es ist weiters möglich das Datumsformat in verkürzter Form „JJJJMM" anzugeben. In diesem Fall wird der Monatsultimo ermittelt.

Beispiel:

<IS153200001 MP=201312> < IS153300001 wird in

<IS153200001 MP=20131231> < IS153300001 transformiert

Weitere Formate für die Eingabe von Meldeperioden in Beispielform sind (welche auch auf den jeweiligen Monatsultimo transformiert werden):

Eingabe	Wird transformiert in
2013J	20131231
201401H	20140630
201501Q	20150331
201601M	20160131
201601	20160131

Variable Meldeperiodenangabe

Um die Anzahl der Prüfregeln zu minimieren und nicht für jede Meldeperiode neue Prüfregeln anzulegen, können Prüfungen mit Vorperioden mit variablen (relativer) Meldeperiodenangaben erzeugt werden.

Beispiel:

<IS153200001 MP=-1M> < IS153300001

Die oberhalb angeführte Prüfregel überprüft die Werte des gleichen Konzepts zu unterschiedlichen Perioden, wobei hier der Wert des Vormonats strikt kleiner sein muss als in der jetzigen Periode.

Die anschließende Tabelle illustriert alle variablen Meldeperiodenangaben. ($n \in \mathbb{N}: n < 100$)

Eingabe	Beschreibung
-nT	Rechnet n Tagesmeldungsperioden zurück
-nM	Rechnet n Monatsmeldungsperioden zurück
-nQ	Rechnet n Quartalsmeldungsperioden zurück

-nH	Rechnet n Halbjahresmeldungsperioden zurück
-nJ	Rechnet n Jahresmeldungsperioden zurück
-nP	Rechnet n Meldungsperioden anhand der Periodizität der Erhebung zurück (z.B. Monate bei einer Erhebung die Monatlich zu melden ist)

Ein Zusammensetzen dieser Variablen ist auch möglich.

Beispiel:

<IS153200001 MP=-1J4Q>

setzt sich auf den Vorjahresultimo ausgehend von der Ankerperiode.

Variable Meldeperioden und Gruppierungen

Bei der Verwendung von Gruppierungen und variablen Meldeperioden ist zu beachten, dass von der Berechnungsperiode aus jede variable Meldeperiode eruiert wird, und danach die für die eruierte Meldeperiode richtige Gruppierung zur Berechnung der Rechenregel herangezogen wird.

2.3.5 Aggregatsfunktion

Es gibt die Möglichkeit Funktionen im Dimensionsaufruf zu verwenden (Liste siehe Anhang „Aggregatsfunktionen“). Diese sind bei der Dimensionszuweisung mit einem „/“ anzufügen.

Beispiel:

<RR11135917 LD_G=SEKTOR_EU/X>

<IS1123581321 WG_G=[EUR, USD, GBP]/S >

Bestandteile mit mehreren Dimensionen können auch einzeln mit einer Aggregatsfunktion aufgelöst werden. Die Funktionen werden von der letzten Dimension (mit einer Funktion) gestartet und ausgewertet. Dies kann bei einem Bestandteil mit *n* Dimensionen als n-dimensionale Matrix gesehen werden. Die Reihenfolge der Dimensionen des Bestandteiles gibt somit den Ablauf der Auflösung der Matrix an.

***Beispiel 1a**: Ablauf der Auflösung von <IS01 WG_G=ALLE/X LD_G=ALLE/S>:*

*<**IS01** WG_G=ALLE/X LD_G=ALLE/S>*

IS01	AUT	GER	GBR
EUR	1	2	3
GBP	5	4	6
USD	4	6	9

*<IS01 WG_G=ALLE/X **LD_G=ALLE/S**>*

Dimensionen werden aufsummiert.

IS01	AUT	GER	GBR	SUMME
EUR	1	2	3	6
GBP	5	4	6	15
USD	4	6	9	19

*<IS01 **WG_G=ALLE/X** LD_G=ALLE/S>*

Anschließend wird das Maximum ermittelt.

IS01	AUT	GER	GBR	SUMME
EUR	1	2	3	6
GBP	5	4	6	15
USD	4	6	9	19

***Beispiel 1b** Ablauf der Auflösung von <IS01 LD_G=ALLE/S WG_G=ALLE/X> (die Komponenten wurden im Vergleich zu 1a vertauscht):*

*<IS01 LD_G=ALLE/S **WG_G=ALLE/X**>*

Das Maximum wird ermittelt.

IS01	AUT	GER	GBR
EUR	1	2	3
GBP	5	4	6
USD	4	6	9
MAX	5	6	9

*<IS01 **LD_G=ALLE/S** WG_G=ALLE/X>*

Anschließend werden die Werte aufsummiert.

IS01	AUT	GER	GBR	SUMME
EUR	1	2	3	
GBP	5	4	6	
USD	4	6	9	
MAX	5	6	9	20

Es können auch Stringwerte mittels dieser Funktion berechnet werden. Eine genauere Erklärung ist in Kapitel 0 vorhanden.

2.4 Syntax-Kommentare

Zusätzliche zu den oben erwähnten Möglichkeiten eine Prüfregelsyntax zu erstellen, können Kommentare in einer Syntax vorkommen. Da es sich bei Kommentaren um Annotationen handelt, werden diese bei der Verarbeitung der Prüfregelsyntax ignoriert und haben keinen Einfluss auf das Ergebnis.

Es gibt zwei Möglichkeiten die Prüfregelsyntax zu kommentieren:

- Block-Kommentar, mit /* als Anfang und */ als Ende. Der Block-Kommentar kann auch mehrzeilig sein.

  Beispiel: <ISKBSB330013 / ein Kommentar innerhalb der Syntax */ LD_G=BSEU/S WG=PLN>*

- Zeilen-Kommentare:

  Beispiel: <ISKBSB330013 LD_G=BSEU/S WG=PLN> //ein Kommentar bis zum Zeilenende

3 Funktionen

Die folgenden Funktionen können in Prüfungen vorkommen:

Name	Gruppe	Aufruf	Beschreibung	Parameter	Beispiel
getDay	time	getDay(x)	Die Funktion ermittelt den Tag eines absoluten oder relativen Datums. Wird kein Datum angegeben, so wird der aktuelle Tag angegeben.	x: Skalar/Vektor/ String	getDay("20120203") -→ 03; Meldestichtagsdatum ist 20120203 getDay("-3D")-→ 31;
getMonth	time	getMonth(x)	Die Funktion ermittelt den Monat eines absoluten oder relativen Datums. Wird kein Datum angegeben, so wird das aktuelle Monat angegeben.	x: Skalar/Vektor/ String	getMonth("20120203") -→ 02; Meldestichtagsdatum ist 201202 getMonth("-3M") -→ 11;
getYear	time	getYear(x)	Die Funktion ermittelt das Jahr eines absoluten oder relativen Datums. Wird kein Datum angegeben, so wird das aktuelle Jahr angegeben.	x: Skalar/Vektor/ String,	getYear("20120203") -→ 2012; Meldestichtagsdatum ist 20120203 getYear("-3J") -→ 2009;
mean	math	mean(x)	Berechnet den Mittelwert des Vektors x. Es können auch mehrere („;“) Vektoren oder Skalare ausgewertet werden. (Möglichkeit "NA" zu ignorieren; Mit min(x, "ignoreNa")).	x: Vektor	x=[1, 2, 3, 4, 5, 6, NA, NA]; mean(x) → techn. Fehler; mean(x, "ignoreNa") → 3.5; mean(NA, NA, "ignoreNa") → techn. Fehler

median	math	median(x)	Berechnet den Median des Vektors x.
			(Möglichkeit "NA" zu ignorieren; Mit median(x, "ignoreNa", "lastNa", "firstNa"); Wobei mit "lastNa" die NAs am Schluss des Vektors gereiht werden (NaNs werden hier wie NA behandelt und daher auch in NA umgewandelt). Mit "firstNa" werden die NAs an den Anfang des Vektors gereiht (NaNs werden hier auch wie NA behandelt und daher auch in NA umgewandelt)).
				x: Vektor	x=[0, 1, 2, 3, 4, 5, 6, 0, 0];
					median(x) → 2;
					y = [NA, 1, 2, 3, 4, 5, 6, NA, NA];
					median(x, "ignoreNa") → 3.5;
					median(NA, NA, "ignoreNa") → techn. Fehler

mod	math	mod(x, y)	Berechnet den ganzzahligen Rest der Division x/y.	x, y: Skalar	x = 5, y = 2; mod(x, y) → 1;

round	math	round(x, y, z)	Rundet die Zahl x auf die Stelle y (wenn negative Zahl, auf 10er, 100er, 1000er, usw. runden, falls positiv, auf Nachkommastellen runden) mittels Verfahren z (1 mathematisch, 2 abrunden, 3 aufrunden). Der Parameter ist optional, falls der Parameter nicht angegeben wird, wird defaultmäßig das Verfahren 1 – mathematisch verwendet.	x: Skalar/Vektor
				y, z: Skalar
					x = 1234.57, y = -1;
					round(x, y, 1) → 1230;
					x = 1234.57, y = 1;
					round(x, y, 1) → 1234.6;

trunc	math	trunc(x, y)	Schneidet die Zahl x bis zur y. Nachkommastelle ab. (y kann positiv und negativ sein.).	x: Skalar/Vektor
				y: Skalar
					x = 1234.3456, y = 2;
					trunc(x, y) → 1234.34;
					x = 1234.3456, y = -1;
					trunc(x, y) → 1230;

concat	misc	concat(x, y,:)	Fügt die Vektoren zu einem gemeinsamen String zusammen.	x, y: Skalar/Vektor	x="20101231", y=0.95; concat(x, "_", y) → "20101231_0.95";
if	misc	if(logischer Wert, Anweisung1 , Anweisung2)	If-Bedingung, ist „logischer Wert" True, dann führe „Anweisung1" aus, ansonsten führe „Anweisung2" aus. Falls der „logischer Wert" ein Vektor ist wird die If-Abfrage für das erste Element des Vektors ausgeführt und das Ergebnis für alle Vektoren berechnet. Unterstützt werden boolsche True und False, 0, 1, und string TRUE und FALSE (caseinsensitiv).	x: Vektor, Skalar	x=[4, 5, 6]; if(x>5, x=0, x=1) → x=[1, 1, 1];
ifElse	misc	ifElse(logischer Wert, Anweisung1 , Anweisung2)	If-Bedingung, ist „logischer Wert" True, dann führe „Anweisung1" aus, ansonsten führe „Anweisung2" aus. Falls der „logischer Wert" ein Vektor ist wird die If-Abfrage für jedes Element des Vektors ausgeführt. Unterstützt werden boolsche True und False, 0, 1, und string TRUE und FALSE (caseinsensitiv).	x: Vektor, Skalar	x=[6, 5, 6]; ifElse(x>5, x=0, x=1) → x=[0, 1, 0];

3.1 Mathematische Funktionen

Alle mathematischen Funktionen werden in Prüfungen verwendet.

3.1.1 multDiffVector

(Hochrechnungsfunktion)

multDiffVector multipliziert zwei Konzepte miteinander, wobei das zweite Konzept weniger Dimensionen besitzen kann als das erste Konzept. Jedoch darf das zweite Konzept nur eine Teilmenge an Dimensionen des ersten Konzeptes besitzen. Diese Funktion kann für die Hochrechnung von Werten verwendet werden.

Diese Funktion ist nur für numerische Werte umgesetzt. Alle anderen Datentypen generieren einen Fehler.

Funktionsaufruf:

multDiffVector(#Param1, # Param2)

Beispiel:

multDiffVector (<IS01>, <IS02>)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept oder Rechenregel

Parameter #Param2 ist ein IS-Konzept, OS-Konzept oder Rechenregel

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA
#NR	techn. Fehler
Number as String	techn. Fehler

Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Beispiel:

Annahme:

IS01:

MO	LD	Wert
191	AT	100
221	AT	100
358	DE	100
358	AT	100

IS02:

MO	Wert
191	2
221	3
358	4

Aufruf:

multDiffVector(<IS01>, <IS02>)

Ergebnis:

MO	LD	Wert
191	AT	200
221	AT	300
358	DE	400
358	AT	400

3.1.2 pow

Die Funktion pow gibt den Wert von #Param1 als Potenz von #Param2 zurück.

Funktionsaufruf:

pow(#Param1, #Param2)

Beispiel:

pow(<IS01>, 2)

<IS01>^2

Parameterbeschreibung:

Parameter #Param1 und #Param2 ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

#Param1 ist die Basis.

#Param2 ist der Exponent.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu techn. Fehler (zu wenig Parameter) ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge führt zu einem techn. Fehler ignoreNa = False → leere Menge wird zu NA

Beispiele:

Ausgangskonzepte:

<IS01>

MO	LD	Wert
1200MO	AT	2
1200MO	DE	4
1200MO	NL	NA
3400MO	AT	8

<IS02>

MO	LD	Wert
1200MO	AT	2
1200MO	DE	0
1200MO	NL	12
3400MO	AT	3

<IS03>

MO	LD	Wert

Beispiel 1:

Aufruf:

pow(<IS01>, <IS02>)

Ergebnis:

MO	LD	Wert
1200MO	AT	4
1200MO	DE	1
1200MO	NL	NA
3400MO	AT	512

Beispiel 2:

Aufruf:

pow(<IS01>, <IS02>, "ignoreNa")

Ergebnis:

Technischer Fehler

Beispiel 3:

Aufruf:

pow(<IS01>, 2)

Ergebnis:

MO	LD	Wert
1200MO	AT	4
1200MO	DE	16
1200MO	NL	NA
3400MO	AT	64

Beispiel 4:

Aufruf:

<IS03>^<IS03>

Ergebnis:

MO	LD	Wert

Beispiel 5:

Aufruf:

<IS01>^<IS03>)

Ergebnis:

MO	LD	Wert
1200MO	AT	NA
1200MO	DE	NA
1200MO	NL	NA
3400MO	AT	NA

Beispiel 6:

Aufruf:

pow<IS01>,<IS03>, "ignoreNa")

Ergebnis:

Technischer Fehler

Beispiel 7:

Aufruf:

pow(<IS03>,<IS03>, "ignoreNa")

Ergebnis:

MO	LD	Wert

Weitere Beispiele:

Aufruf	Ergebnis
pow(2, NA)	NA
pow (NA, 2)	NA
pow(2, NA, "ignoreNa")	techn. Fehler: Wrong number of parameters (2 expected)
pow (NA, 2, "ignoreNa")	techn. Fehler: Wrong number of parameters (2 expected)
pow("test", 2)	techn. Fehler
pow(2, "test")	techn. Fehler
pow(2, True)	techn. Fehler
pow(True, 2)	techn. Fehler
pow(NaN, 2)	NaN
pow(2, NaN)	NaN

3.1.3 unaryMinus

Die Funktion negiert die mitgelieferte numerischen Variable.

Funktionsaufruf:

unaryMinus(#Param1)

Beispiel:

unaryMinus(<IS01>)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel oder ein Eingabewert.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu leere Menge ignoreNa = False → leere Menge wird zu leere Menge

Beispiele:

Ausgangskonzepte:

<IS01>

MO	Wert
1200MO	-12
3400MO	123
5600MO	0
9900MO	NA
1100MO	NaN

<IS02>

MO	Wert
1200MO	-12
3400MO	#NR

Beispiel 1:

Aufruf:

unaryMinus(<IS01>)

Ergebnis:

MO	Wert
1200MO	12
3400MO	-123
5600MO	0
9900MO	NA
1100MO	NaN

Beispiel 2:

Aufruf:

unaryMinus(<IS02>)

Ergebnis:

techn. Fehler

3.1.4 errDiv

Funktionsbeschreibung

Es wird die Division #Param1 / #Param2 ausgeführt, sollte es dabei zu einer Division durch 0 kommen, wird als Ergebnis der Wert von #Param3 ausgegeben.

Funktionsaufruf:

errDiv(#Param1, #Param2, #Param3)

Beispiel:

errDiv(<IS01>, <IS02>, <IS03>)

Parameterbeschreibung

#Param1, #Param2, #Param3 ist ein IS-Konzept, OS-Konzept, eine Rechenregel oder ein Eingabewert

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Beispiele

Ausgangskonzept:

<IS01>

MO	LD	Werte
1200MO	AT	100
3400MO	DE	600
5600MO	DE	500
5600MO	AT	200

<IS02>

MO	LD	Werte
1200MO	AT	0
3400MO	DE	300
5600MO	DE	1000
5600MO	AT	100

Beispiel:

Aufruf:

errDiv(<IS01>,<IS02>,50)

Ergebnis:

MO	LD	Werte
1200MO	AT	50
3400MO	DE	2
5600MO	DE	0,5
5600MO	AT	2

3.2 Mathematische Funktionen - Intervallarithmetik

3.2.1 multDiffVector – Intervallarithmetik

Funktionsbeschreibung

multDiffVector multipliziert zwei Konzepte nach den Regeln der Intervallarithmetik miteinander, wobei das zweite Konzept weniger Dimensionen besitzen kann als das erste Konzept. Jedoch darf das zweite Konzept nur eine Teilmenge an Dimensionen des ersten Konzeptes besitzen. Diese Funktion kann für die Hochrechnung von Werten verwendet werden.

Funktionsaufruf

multDiffVector(Param1, Param2)

Beispiel:

multDiffVector (<IS01>, <IS02>)

Parameterbeschreibung

Parameter #Param1 ist ein IS-Konzept, OS-Konzept oder Rechenregel

Parameter #Param2 ist ein IS-Konzept, OS-Konzept oder Rechenregel

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler

Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Beispiele

Ausgangskonzepte:

<IS01>, @decimals=-3

MO	LD	WG	Werte
1200MO	AT	EUR	1000
1200MO	DE	EUR	6000
1200MO	IT	EUR	5000
1200MO	US	USD	2000

<IS02>, @decimals=-3

MO	LD	Werte
1200MO	AT	5000
1200MO	DE	3000
1200MO	IT	10000
1200MO	US	1000

Beispiel 1

Aufruf:

multDiffVector (<IS01>, <IS02>)

Ergebnis:

MO	LD	WG	Werte
1200MO	AT	EUR	[2250000; 8250000]
1200MO	DE	EUR	[13750000; 22750000]
1200MO	IT	EUR	[42750000; 57750000]

1200MO	US	USD	[750000; 3750000]

Beispiel 2

Aufruf:

multDiffVector (<IS01>, <IS02>, "ignoreInterval")

Ergebnis:

MO	LD	WG	Werte
1200MO	AT	EUR	5000000
1200MO	DE	EUR	18000000
1200MO	IT	EUR	50000000
1200MO	US	USD	2000000

3.2.2 errDiv – Intervallarithmetik

Funktionsbeschreibung

errDiv dividiert zwei Konzepte nach den Regeln der Intervallarithmetik, wobei bei Division durch ein Intervall, das 0 beinhaltet, das dritte angegebene Konzept ausgegeben werden soll.

Funktionsaufruf

errDiv(Konzept1, Konzept2, Konzept3)

Beispiel:

errDiv(<IS01>, <IS02>, <IS03>)

Parameterbeschreibung

- Konzept1 ist ein IS-Konzept, OS-Konzept oder Rechenregel
- Konzept2 ist ein IS-Konzept, OS-Konzept oder Rechenregel
- Konzept3 ist ein IS-Konzept, OS-Konzept oder Rechenregel

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Algorithmus

errDiv dividiert zwei Konzepte nach den Regeln der Intervallarithmetik. Bei einer Division durch ein Intervall das 0 beinhaltet, wird der Wert aus dem dritten Parameter ausgegeben.

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

<IS02>, @decimals=-3

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	10000
5600MO	AT	1000

<IS03>, @decimals=-3

MO	LD	Werte
1200MO	AT	500
3400MO	DE	3000
5600MO	DE	0
5600MO	AT	1000

Beispiel 1

Aufruf:

errDiv(<IS01>,<IS02>,<IS03>)

Ergebnis:

MO	LD	Werte
1200MO	AT	[0,090909091; 0,333333333]
3400MO	DE	[1,571428571; 2,6]
5600MO	DE	[0,428571429; 0,578947368]
5600MO	AT	[1; 5]

Beispiel 2

Aufruf:

errDiv(<IS01>,<IS02>,<IS03>, "ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	[0,2; 0,2]
3400MO	DE	[2; 2]
5600MO	DE	[0,5; 0,5]
5600MO	AT	[2; 2]

Beispiel 3

Aufruf:

errDiv(<IS01>,<IS03>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	[4500; 5500]*
3400MO	DE	[1,571428571; 2,6]
5600MO	DE	[9500; 10500]*
5600MO	AT	[1; 5]

*0 im Intervall des Divisiors

Beispiel 4

Aufruf:

errDiv(<IS01>,<IS03>,<IS02>, "ignoreInterval")

Ergebnis:

MO	LD	Werte
1200MO	AT	[2; 2]
3400MO	DE	[2; 2]
5600MO	DE	[10000; 10000]
5600MO	AT	[2; 2]

3.3 Logische-Funktionen

3.3.1 and

Die and-Funktion liefert ein True zurück, wenn alle Werte True sind. Wenn ein oder mehr Werte False sind, ist das Ergebnis False. Numerische Werte werden in boolsche Werte gecastet (0 wird zu False und numerischen Werte ungleich 0 zu True).

Funktionsaufruf:

and(#ParamN)

Beispiel:

and(<IS01>, <IS02>)

<IS01>&&<IS02>

Parameterbeschreibung:

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Boolean
String	techn. Fehler
Boolean	Boolean
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu False ignoreNa = False → leere Menge wird zu NA

Beispiele:

Ausgangskonzepte:

<IS01>

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	NA
3400MO	AT	123

<IS02>

MO	LD	Wert

Beispiel 1:

Aufruf:

and(<IS01>, <IS01>)

Ergebnis:

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	NA
3400MO	AT	True

Beispiel 2:

Aufruf:

and(<IS01>, <IS01>, "ignoreNa")

Ergebnis:

MO	LD	Wert
1200MO	AT	True

1200MO	DE	False
1200MO	NL	False
3400MO	AT	True

Beispiel 3:

Aufruf:

and(<IS01>, <IS02>)

Ergebnis:

MO	LD	Wert
1200MO	AT	NA
1200MO	DE	NA
1200MO	NL	NA
3400MO	AT	NA

Beispiel 4:

Aufruf:

and(<IS01>, <IS02>, "ignoreNa")

Ergebnis:

MO	LD	Wert
1200MO	AT	False
1200MO	DE	False
1200MO	NL	False
3400MO	AT	False

Beispiel 5:

Aufruf:

and(<IS02>, <IS02>)

Ergebnis:

MO	LD	Wert

Beispiel 6:

Aufruf:

and(<IS02>, <IS02>, "ignoreNa")

Ergebnis:

MO	LD	Wert

3.3.2 or

Die or-Funktion liefert ein True zurück, wenn mindestens ein Wert True ist. Wenn alle Werte False sind, ist das Ergebnis False. Numerische Werte werden in boolsche Werte gecastet (0 wird zu False und numerischen Werte ungleich 0 zu True).

Funktionsaufruf:

or(#ParamN)

Beispiel:

or(<IS01>, <IS02>)

<IS01>||<IS02>

Parameterbeschreibung:

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Boolean
String	techn. Fehler
Boolean	Boolean
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu False ignoreNa = False → leere Menge wird zu NA

Beispiele:

Ausgangskonzepte:

<IS01>

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	NA
3400MO	AT	123

<IS02>

MO	LD	Wert
1200MO	AT	False
1200MO	DE	False
1200MO	NL	456
3400MO	AT	0

<IS03>

MO	LD	Wert

Beispiel 1:

Aufruf:

or(<IS01>, <IS02>)

Ergebnis:

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	NA
3400MO	AT	True

Beispiel 2:

Aufruf:

or(<IS01>, <IS02>, "ignoreNa")

Ergebnis:

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	True
3400MO	AT	True

Beispiel 3:

Aufruf:

or(<IS01>, <IS03>)

Ergebnis:

MO	LD	Wert
1200MO	AT	NA
1200MO	DE	NA
1200MO	NL	NA
3400MO	AT	NA

Beispiel 4:

Aufruf:

or(<IS01>, <IS03>, "ignoreNa")

Ergebnis:

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	False
3400MO	AT	True

Beispiel 5:

Aufruf:

or(<IS03>, <IS03>)

Ergebnis:

MO	LD	Wert

Beispiel 6:

Aufruf:

or(<IS03>, <IS03>, "ignoreNa")

Ergebnis:

MO	LD	Wert

3.3.3 xor

Die xor-Funktion liefert ein True zurück, wenn ein Wert True und ein Wert False ist. Wenn alle Werte True oder alle Werte False sind wird ein False zurück geliefert.

Numerische Werte werden in boolsche Werte gecastet (0 wird zu False und numerischen Werte ungleich 0 zu True).

Funktionsaufruf:

xor(#ParamN)

Beispiel:

xor(<IS01>, <IS02>)

!xor(<IS01>, <IS02>) → (äquivalent zu xnor(<IS01>, <IS02>))

Parameterbeschreibung:

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Boolean
String	techn. Fehler
Boolean	Boolean
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu False ignoreNa = False → leere Menge wird zu NA

Beispiele:

Ausgangskonzepte:

<IS01>

MO	LD	Wert
1200MO	AT	True
1200MO	DE	True
1200MO	NL	NA
3400MO	AT	123

<IS02>

MO	LD	Wert
1200MO	AT	False
1200MO	DE	True
1200MO	NL	456
3400MO	AT	0

<IS03>

MO	LD	Wert

Beispiel 1:

Aufruf:

xor(<IS01>, <IS02>)

Ergebnis:

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	NA

3400MO	AT	True

Beispiel 2:

Aufruf:

xor(<IS01>, <IS02>, "ignoreNa")

Ergebnis:

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	True
3400MO	AT	True

Beispiel 3:

Aufruf:

xor(<IS01>, <IS03>)

Ergebnis:

MO	LD	Wert
1200MO	AT	NA
1200MO	DE	NA
1200MO	NL	NA
3400MO	AT	NA

Beispiel 4:

Aufruf:

xor(<IS01>, <IS03>, "ignoreNa")

Ergebnis:

MO	LD	Wert
1200MO	AT	True

1200MO	DE	True
1200MO	NL	False
3400MO	AT	True

Beispiel 5:

Aufruf:

xor(<IS03>, <IS03>)

Ergebnis:

MO	LD	Wert

Beispiel 6:

Aufruf:

xor(<IS03>, <IS03>, "ignoreNa")

Ergebnis:

MO	LD	Wert

3.3.4 xnor

Die xnor-Funktion liefert ein True zurück, wenn alle Teilaussagen True oder alle Teilaussagen False sind. (Die xnor-Funktion ist eine negierte xor-Funktion.)

Numerische Werte werden in boolsche Werte gecastet (0 wird zu False und numerischen Werte ungleich 0 zu True).

Funktionsaufruf:

xnor(#ParamN)

Beispiel:

xnor(<IS01>, <IS02>)

!xnor(<IS01>, <IS02>) → (äquivalent zu xor(<IS01>, <IS02>))

Parameterbeschreibung:

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Boolean
String	techn. Fehler
Boolean	Boolean
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu False ignoreNa = False → leere Menge wird zu NA

Beispiele:

Ausgangskonzepte:

<IS01>

MO	LD	Wert
1200MO	AT	True
1200MO	DE	True
1200MO	NL	NA
3400MO	AT	123

<IS02>

MO	LD	Wert
1200MO	AT	False
1200MO	DE	True
1200MO	NL	0
3400MO	AT	0

<IS03>

MO	LD	Wert

Beispiel 1:

Aufruf:

xnor(<IS01>, <IS02>)

Ergebnis:

MO	LD	Wert
1200MO	AT	False
1200MO	DE	True
1200MO	NL	NA

3400MO	AT	False

Beispiel 2:

Aufruf:

xnor(<IS01>, <IS02>, "ignoreNa")

Ergebnis:

MO	LD	Wert
1200MO	AT	False
1200MO	DE	True
1200MO	NL	True
3400MO	AT	False

Beispiel 3:

Aufruf:

xnor(<IS01>, <IS03>)

Ergebnis:

MO	LD	Wert
1200MO	AT	NA
1200MO	DE	NA
1200MO	NL	NA
3400MO	AT	NA

Beispiel 4:

Aufruf:

xnor(<IS01>, <IS03>, "ignoreNa")

Ergebnis:

MO	LD	Wert
1200MO	AT	False
1200MO	DE	False

1200MO	NL	True
3400MO	AT	False

Beispiel 5:

Aufruf:

xnor(<IS03>, <IS03>)

Ergebnis:

MO	LD	Wert

Beispiel 6:

Aufruf:

xnor(<IS03>, <IS03>, "ignoreNa")

Ergebnis:

MO	LD	Wert

3.3.5 *isApprox*

Die Funktion isApprox überprüft, ob #Param1 und #Param2 annähernd (Grenze #Param3) gleich sind. Die Funktion liefert ein True, wenn der Absolutwert der Differenz zwischen #Param1 und #Param2 kleiner gleich der Grenze #Param3 (|#Param1 - #Param2| <= #Param3) ist. Andernfalls wird False zurückgegeben.

Funktionsaufruf:

isApprox(#Param1, #Param2, #Param3)

Beispiel:

isApprox(10, 12, 2)

Parameterbeschreibung:

Parameter #Param1, #Param2 und #Param3 sind ein IS-Konzept, OS-Konzept, Rechenregel oder Eingabewert.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 gecastet → Ergebnis hängt von den Ausgangswerten ab ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge (1 od. 2 Parameter)	ignoreNa = True → leere Menge wird zu NA und NA zu 0 → Ergebnis hängt von den Ausgangswerten ab ignoreNa = False → leere Menge wird zu NA

Leere Menge (alle Parameter)	Leere Menge

Beispiele:

Ausgangskonzepte:

<IS01>

MO	Wert
1200MO	0
3400MO	15
5600MO	20
7800MO	25
2200MO	30

<IS02>

MO	Wert
1200MO	0
3400MO	20
5600MO	40
7800MO	60
2200MO	NA

<IS03>

MO	LD	Wert

Beispiel 1:

Aufruf

isApprox(<IS01>,<IS02>,10)

Ergebnis

MO	Wert
1200MO	True

3400MO	True
5600MO	False
7800MO	False
2200MO	NA

Beispiel 2a:

Aufruf

isApprox(<IS01>, <IS02>, <IS03>)

Ergebnis

MO	Wert
1200MO	NA
3400MO	NA
5600MO	NA
7800MO	NA
2200MO	NA

Beispiel 2b:

Aufruf

isApprox(<IS01>, <IS02>, <IS03>, "ignoreNa")

Ergebnis

MO	Wert
1200MO	True
3400MO	False
5600MO	False
7800MO	False
2200MO	False

Beispiel 3a:

Aufruf

isApprox(<IS01>, <IS03>, <IS02>)

Ergebnis

MO	Wert
1200MO	NA
3400MO	NA
5600MO	NA
7800MO	NA
2200MO	NA

Beispiel 3b:

Aufruf

isApprox(<IS01>, <IS03>, <IS02>, "ignoreNa")

Ergebnis

MO	Wert
1200MO	True
3400MO	True
5600MO	True
7800MO	True
2200MO	False

Beispiel 4a:

Aufruf

isApprox(<IS03>, <IS02>, <IS01>)

Ergebnis

MO	Wert
1200MO	NA
3400MO	NA

5600MO	NA
7800MO	NA
2200MO	NA

Beispiel 4b:

Aufruf

isApprox(<IS03>, <IS02>, <IS01>, "ignoreNa")

Ergebnis

MO	Wert
1200MO	True
3400MO	False
5600MO	False
7800MO	False
2200MO	True

Weitere Beispiele:

Aufruf	Ergebnis
isApprox(2, 5, NA)	NA
isApprox(NA, 2, 5)	NA
isApprox(2, 3, NA, "ignoreNa")	False
isApprox(2, NA, 3, "ignoreNa")	True
isApprox(NA, 3, 1, "ignoreNa")	False

3.3.6 allUnique

Die Funktion allUnique gibt ein True zurück, wenn kein Wert doppelt vorkommt. Ansonsten wird ein False zurückgegeben.

Funktionsaufruf:

allUnique(#ParamN)

Beispiel:

allUnique(<IS01>)

Parameterbeschreibung:

Parameter #ParamN ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder String.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Boolean
String	Boolean
Boolean	techn. Fehler
Date	Boolean
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 → Ergebnis wird ein Boolean (außer, wenn ausschließlich NAs vorkommen → NA wird zu NA) ignoreNa = False → NA wird zu NA
#NR	Boolean
Number as String	Boolean
Boolean as String	Boolean
Date as String	Boolean
Leere Menge (nicht alle Parameter)	leere Menge wird aufgespannt mit NA ignoreNa = True → NA wird zu 0 → Ergebnis wird ein Boolean ignoreNa = False → leere Menge wird zu NA

Leere Menge (2 bis n Parameter)	Leere Menge
Leere Menge (1 Parameter)	NA

Beispiele:

Ausgangskonzepte:

<IS01>

MO	LD	Wert
1200MO	AT	5
1200MO	DE	2
1200MO	PL	9
1400MO	US	1
1400MO	UK	8
1400MO	AT	98
5300MO	DE	8
5300MO	SK	4
5300MO	EE	8
5300MO	PR	6
5600MO	AT	3
5600MO	DE	2
5600MO	PL	9

<IS02>

MO	LD	Wert

Beispiel 1:

Aufruf

allUnique(<IS02>, <IS02>)

Ergebnis

MO	LD	Wert

Beispiel 2:

Aufruf

allUnique(<IS02>, <IS02>, "ignoreNa")

Ergebnis

MO	LD	Wert

Weitere Beispiele:

Aufruf	Ergebnis
allUnique(<IS01>)	False
allUnique(<IS01 MO=1400MO>)	True
allUnique(<IS01 MO=5300MO>)	False
allUnique(getDim(<IS01>, LD))	False
allUnique(getDim(<IS01 MO=1200MO>, LD))	True
allUnique(<IS02>)	NA
allUnique(<IS02>, "ignoreNa")	NA
allUnique(NA, 3)	NA
allUnique(NA, 3, "ignoreNa")	True
allUnique(NA, 0, "ignoreNa")	False
allUnique(NA, NA, "ignoreNa")	NA

3.4 Logische-Funktionen - Intervallarithmetik

3.4.1 and – Intervallarithmetik

Funktionsbeschreibung

Die and-Funktion liefert ein True zurück, wenn alle Werte True sind. Wenn ein oder mehr Werte False sind, ist das Ergebnis False. Numerische Werte werden in boolsche Werte gecastet (0 wird zu False und numerischen Werte ungleich 0 zu True).

Funktionsaufruf

and(#ParamN)

Beispiel:

and(<IS01>, <IS02>)

<IS01>&&<IS02>

Parameterbeschreibung

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Boolean
String	techn. Fehler
Boolean	Boolean
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu False

	ignoreNa = False → leere Menge wird zu NA

Beispiele

Ausgangskonzepte

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	1500

<IS02>, @decimals=-3

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	4700
5600MO	AT	1000

Beispiel 1

Aufruf:

and(<IS01> > <IS02>, <IS01> <= <IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	False
3400MO	DE	False
5600MO	DE	True
5600MO	AT	True

3.4.2 or – Intervallarithmetik

Funktionsbeschreibung

Die or-Funktion liefert ein True zurück, wenn mindestens ein Wert True ist. Wenn alle Werte False sind, ist das Ergebnis False. Numerische Werte werden in boolsche Werte gecastet (0 wird zu False und numerischen Werte ungleich 0 zu True).

Funktionsaufruf:

or(#ParamN)

Beispiel:

or(<IS01>, <IS02>)

<IS01>||<IS02>

Parameterbeschreibung

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Boolean
String	techn. Fehler
Boolean	Boolean
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu False ignoreNa = False → leere Menge wird zu NA

Beispiele

Ausgangskonzepte

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	1500

<IS02>, @decimals=-3

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	4700
5600MO	AT	1000

Beispiel 1

Aufruf

or(<IS01> > <IS02>, <IS01> + <IS02> > 10000)

Ergebnis

MO	LD	Werte
1200MO	AT	False
3400MO	DE	True
5600MO	DE	True
5600MO	AT	True

3.4.3 ifElse – Intervallarithmetik

Funktionsbeschreibung

Die ifElse-Funktion führt Code abhängig vom Wahrheitswert von #Param1 aus. Ist dieser wahr, wird #Param2 rückgegeben. Anderenfalls wird #Param3 rückgegeben.

Funktionsaufruf

ifElse(#Param1, #Param2, #Param3)

Beispiel:

ifElse(<IS01>,<IS02>,<IS03>)

Parameterbeschreibung

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Boolean
String	techn. Fehler
Boolean	Boolean
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu False ignoreNa = False → leere Menge wird zu NA

Beispiele

Ausgangskonzepte

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	1500

<IS02>, @decimals=-3

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	4700
5600MO	AT	1000

Beispiel 1

Aufruf:

ifElse(<IS01> > <IS02>, <IS01>, 0)

Ergebnis:

MO	LD	Werte
1200MO	AT	[0;0]
3400MO	DE	[5500; 6500]
5600MO	DE	[4500; 5500]
5600MO	AT	[1000; 2000]

Beispiel 2

ifElse(<IS01> <= <IS02>, "left", NA)

Ergebnis:

MO	LD	Werte
1200MO	AT	left
3400MO	DE	NA
5600MO	DE	left
5600MO	AT	left

3.5 Date-Funktionen

Alle Date-Funktionen werden in Prüfungen verwendet.

3.5.1 currentDate

Die Funktion currentDate gibt das aktuelle Datum bzw. den Datenstand der Aktualisierung zurück.

Funktionsaufruf:

currentDate()

Beispiel:

currentDate()

Parameterbeschreibung:

kein Parameter: liefert das aktuelle Datum zurück

DATENSTAND: liefert das Datum, zu dem die Berechnung gestartet worden ist zurück

Datentypkompatibilität:

Datentyp	Ergebnis
Number	technischer Fehler
String	technischer Fehler
Boolean	technischer Fehler
Date	technischer Fehler
NaN	technischer Fehler
NA	technischer Fehler
#NR	technischer Fehler
Number as String	technischer Fehler
Boolean as String	technischer Fehler
Date as String	technischer Fehler
Leere Menge	technischer Fehler

Beispiele:

Beispiel 1:

Aufruf:

currentDate()

Ergebnis:

liefert das heutige Datum im Format YYYYMMDD zurück

Beispiel 2:

Zweck: Aktualisierung des OSDATUM mit RRDATUM für MP=Q4 2017 am 05.12.2018

Aufruf

concat(

getYear(currentDate()),

ifElse(getMonth(currentDate())<10,concat("0",getMonth(currentDate())),getMonth(currentDate())),

ifElse(getDay(currentDate())<10,concat("0",getDay(currentDate())),getDay(currentDate()))

)

Ergebnis

KO	MP	Wert
OSDATUM	20171231	20181205

3.5.2 *dateDiff*

dateDiff subtrahiert zwei Datumswerte voneinander und berechnet die Anzahl der Tage, das Von-Datum ist inkludiert. Das Ergebnis kann auch negativ sein. Es wird weiters zwischen den zwei Berechnungsmöglichkeiten unterschieden:

- BAT - Bankarbeitstage auf Basis des österreichischen Kalenders
  - Manche Feiertage sind fix, andere haben einen Offset zu Ostern. Ostern wird mit der Gauß'schen Osterformel errechnet.
    - 01.01 Neujahr (New Year)
    - 06.01 Hl. Drei Könige (Epiphany)
    - Ostersonntag (Easter Sunday)
    - <Ostern+1> Ostermontag (Easter Monday)
    - 01.05 Staatsfeiertag (national holiday)
    - <Ostern+39> Christi Himmelfahrt (Donnerstag) (Ascension Day)
    - <Ostern+50> Pfingstmontag (Pentecost or Whit Monday)
    - <Ostern+60> Fronleichnam (Donnerstag) (Corpus Christi)
    - 15.08 Maria Himmelfahrt (feast of assumption)
    - 26.10 Nationalfeiertag (national holiday)
    - 01.11 Allerheiligen (All Saint's Day)
    - 08.12 Maria Empfängnis
    - 24.12 Bankfeiertag (bank holiday or Christmas Eve)
    - 25.12 Christtag (Christmas)
    - 26.12 Stefanitag (St. Stephens Day)
- KAT - alle Kalendertage

Funktionsaufruf:

dateDiff(#Parameter1, #Parameter2,#Kalender)

Beispiel:

dateDiff(19871215, 19871224, BAT)

Parameterbeschreibung:

1. Parameter #Parameter1 ist ein IS-Konzept, OS-Konzept, Rechenregel oder Eingabewert
2. Parameter #Parameter2 ist ein IS-Konzept, OS-Konzept, Rechenregel oder Eingabewert
3. Parameter #Kalender hat zwei mögliche Ausprägungen: BAT - Bankarbeitstage auf Basis des österreichischen Kalenders, KAT - alle Kalendertage

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number (wenn Eingabe YYYYMMDD, sonst fach. Fehler)

String	fachl. Error
Boolean	fachl. Error
Date	Number
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA
#NR	Tech. Error
Number as String	Number (wenn Eingabe YYYYMMDD, sonst fach. Fehler)
Boolean as String	Tech. Error
Date as String	Tech. Error
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Beispiel:

dateDiff(20191225, 20191226, BAT) → 0

dateDiff(20191225, 20191226, KAT) → 1

3.5.3 eoMonth

eoM ermittelt mittels Parameter YYYY, MM (ohne führende Null, 1 ist Jänner) den richtigen Monatsultimo.

Funktionsaufruf:

eoMonth(#Parameter1, #Parameter2)

Beispiel:

eoMonth(1987, 12)

Parameterbeschreibung:

1. Parameter #Parameter1
   ist ein IS-Konzept, OS-Konzept, Rechenregel oder Eingabewert (Jahr)
2. Parameter #Parameter2
   ist ein IS-Konzept, OS-Konzept, Rechenregel oder Eingabewert (Monat)

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number
String	Tech. Error
Boolean	Tech. Error
Date	Tech. Error
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 ignoreNa = False → NA bleibt NA
#NR	Tech. Error
Number as String	Tech. Error
Boolean as String	Tech. Error
Date as String	Tech. Error
Leere Menge	ignoreNa = True → leere Menge wird zu 0 ignoreNa = False → leere Menge bleibt NA

Beispiel:

eoMonth(2020,2) -→ 29

3.5.4 asDate

asDate wandelt mittels Parameter YYYY, MM (ohne führende Null, 1 ist Jänner) und TT (ohne führende Null) Zahlen in einen Date-Typ um. asDate(YYYY,MM,TT) -⟶ YYYYMMTT

Funktionsaufruf:

asDate(#Parameter1, #Parameter2,#Parameter3)

Beispiel:

asDate(1987, 12,15)

Parameterbeschreibung:

1. Parameter #Parameter1
   ist ein IS-Konzept, OS-Konzept, Rechenregel oder Eingabewert (Jahr)
2. Parameter #Parameter2
   ist ein IS-Konzept, OS-Konzept, Rechenregel oder Eingabewert (Monat)
3. Parameter #Parameter3
   ist optional und ein IS-Konzept, OS-Konzept, Rechenregel oder Eingabewert (Tag)

Datentypkompatibilität:

Datentyp	Ergebnis
Number	YYYYMMTT (Number)
String	Tech. Error
Boolean	Tech. Error
Date	Tech. Error
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 ignoreNa = False → NA bleibt NA
#NR	Tech. Error
Number as String	Tech. Error
Boolean as String	Tech. Error
Date as String	Tech. Error
Leere Menge	ignoreNa = True → leere Menge wird zu 0 ignoreNa = False → leere Menge bleibt NA

Beispiel:

asDate(2017,1,31) -⟶ 20170131

3.6 Dimensions-Funktionen

Alle Dimensionsfunktionen werden in Prüfungen verwendet.

3.6.1 getDim

Die Funktion getDim soll als Ergebnis einen Vektor basierend auf dem Parameter 1 liefern. Die Werte des Ergebnisses enthalten die Ausprägungen der Dimension des 2. Parameters.

Funktionsaufruf:

getDim(Konzept, Dimension)

Beispiel:

Annahme:

IS01:

MO	LD	Wert
191	AT	1.000
221	AT	500
358	DE	700

Aufruf:

getDim(<IS01>, LD)

Ergebnis:

MO	LD	Wert
191	AT	AT
221	AT	AT
358	DE	DE

3.6.2 renameDim

Die Funktion soll vom Konzept die KonzeptDimension in die RenameDimension umbenennen.

Funktionsaufruf:

renameDim(Konzept, "KonzeptDimension", "RenameDimension")

Parameterbeschreibung:

1. Parameter Konzept

   ist ein IS-Konzept oder OS-Konzept

2. Parameter KonzeptDimension

   ist eine Dimension vom Konzept

3. Parameter RenameDimension

- ist der Name der Dimension, in welche die KonzeptDimension umbenannt werden soll.

Beispiel 1:

Aufruf:

renameDim(<ISBAMADDP>, "INADD", "INBEZ")

Ergebnis:

Das Konzept <ISBAMADDP> hat die Dimensionen MO, MP, EC, DB, IN, INADD. Die Dimension INADD soll in die Dimension INBEZ umbenannt werden, damit das Konzept <ISBAMADDP> die Dimensionen MO, MP, EC, DB, IN, INBEZ besitzt.
Diese Funktion soll dann zum Beispiel für einen Vergleich verwendet werden mit dem Konzept <ISBAMANTUGB> (MO, MP, EC, DB, IN, INBEZ). Da der Vergleich

<ISBAMADDP> <= <ISBAMANTUGB>

bei der Berechnung zu einem Fehler der Dimensionen führt, soll die Syntax

renameDim(<ISBAMADDP>, INADD, INBEZ) <= <ISBAMANTUGB>

verwendet werden. Durch die Umbenennung führt der Vergleich auf größer zu keinem Fehler.

3.6.3 *mappingDimAttribute*

Mit Hilfe der mappingDimAttribute-Methode kann der Code der Dimensionsattribute eines Konzeptes umbenannt werden.

Funktionsaufruf:

mappingDimAttribute (Konzept, KonzeptDimension, Mapping, Spalte)

Parameterbeschreibung:

1. Parameter Konzept
- ist ein IS-Konzept oder OS-Konzept
2. Parameter KonzeptDimension
- ist die eine Dimension vom Konzept
3. Parameter Mapping
- ist der Name des Mappings
4. Parameter Spalte
- ist ein optionaler Parameter, falls es sich um ein Mapping mit mehreren Spalten handelt
-

Beispiel 1:

Annahme:

MPABC

Ausgangswert	Zielwert
BW	A
MW	B
ZW	C

<ISBAMADDP>

MO	MP	INADD	Wert
1200MO	20171231	BW	3
1300MO	20171231	MW	4
1400MO	21071231	ZW	5

Aufruf:

mappingDimAttribute (<ISBAMADDP>, "INADD", MPABC)

Ergebnis:

MO	MP	INADD	Wert
1200MO	20171231	A	3
1300MO	20171231	B	4
1400MO	21071231	C	5

Beispiel 2:

Annahme:

MPABC

Ausgangswert	ABC	DEF
BW	A	D
MW	B	E
ZW	C	F

<ISBAMADDP>

MO	MP	INADD	Wert
1200MO	20171231	BW	3
1300MO	20171231	MW	4
1400MO	21071231	ZW	5

Aufruf:

mappingDimAttribute (<ISBAMADDP>, "INADD", MPABC,"DEF")

Ergebnis**:**

MO	MP	INADD	Wert
1200MO	20171231	D	3
1300MO	20171231	E	4
1400MO	21071231	F	5

3.7 Dimensions-Funktionen - Intervallarithmetik

3.7.1 getDim – Intervallarithmetik

Funktionsbeschreibung

Die Funktion getDim soll die Ausprägung der Dimension #Dimension in den Wert schreiben.

Funktionsaufruf

getDim(#Vektor, #Dimension)

Beispiel:

getDim(<IS01>, LD)

Parameterbeschreibung

- Parameter #Vektor: Ist ein IS-Konzept, OS-Konzept, Rechenregel oder eine ANUBIS-Syntax die einen Vektor zurückgibt (z.B. ein crossJoin)
- Parameter #Dimension: Dimension des Vektors

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	MP	LD	WG	Werte
1200MO	20231231	AT	EUR	1000
3400MO	20231231	DE	EUR	6000
5600MO	20231231	DE	EUR	5000
5600MO	20231231	AT	EUR	2000

Beispiel 1

Aufruf

getDim(<IS01>, LD)

Ergebnis

MO	MP	LD	WG	Werte
1200MO	20231231	AT	EUR	"AT"
3400MO	20231231	DE	EUR	"DE"
5600MO	20231231	DE	EUR	"DE"
5600MO	20231231	AT	EUR	"AT"

Beispiel 2

Aufruf

getDim(<IS01>, MP)

Ergebnis

MO	MP	LD	WG	Werte
1200MO	20231231	AT	EUR	"20231231"
3400MO	20231231	DE	EUR	"20231231"
5600MO	20231231	DE	EUR	"20231231"
5600MO	20231231	AT	EUR	"20231231"

3.8 Aggregats-Funktionen

3.8.1 aggInc

Die Funktion aggInc summiert von einem Konzept alle angegebenen Dimensionen. In der Wertespalte werden Zahlen und Strings bei der Aggregierung berücksichtigt.

Funktionsaufruf:

aggInc(Param1,ParamN)

Beispiel:

aggInc(<IS01>, MO, MP, EC)

Parameterbeschreibung:

- Param 1 ist ein Konzept (Skalar/Vektor)
- ParamN sind 1 bis N Dimensionen, welche vom Konzept unterstützt werden müssen

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number
String	String
Boolean	Boolean
Date	Date
NaN	NaN
NA	ignoreNa = True → NA wird zu NA *(wenn ausschließlich NAs vorkommen, ansonsten wird NA ignoriert)* ignoreNa = False → NA wird zu NA
#NR	#NR
Number as String	Number as String
Boolean as String	Boolean as String
Date as String	Date as String
Leere Menge	ignoreNa = True → leere Menge wird zu leere Menge ignoreNa = False → leere Menge wird zu leere Menge

Algorithmus:

Die Funktion aggInc hat als ersten Parameter ein Konzept und die nachfolgenden Parameter sind Dimensionen, welche summiert werden.

Bei einem nicht Number-Datentyp-Vektor wird ein Ergebnis geliefert, falls jede Zeile den gleichen String-Wert hat. Dann wird auch im Aggregat der gleiche Wert geliefert, sollten sich die Werte unterscheiden liefert die Funktion ein NA.

Folgende Schreibweisen sind somit äquivalent:

```
aggInc(<IS01>, MO, MP, EC) == <IS01 MO_G=ALLE/S MP_G=ALLE/S
EC_G=ALLE/S>
aggInc(<IS01>, LD, WG) == <IS01 LD_G=ALLE/S WG_G=ALLE/S>
aggInc(<IS01 MO_G=BS0100510/S>, LD, WG) == <IS01 MO_G=BS0100510/S
LD_G=ALLE/S WG_G=ALLE/S>
```

Beispiele:

Ausgangskonzept:

<IS01>

MO	MP	EC	LD	WG	Werte
1200MO	202012	GKE1	AT	EUR	12.000.000
3400MO	202012	GKE1	DE	EUR	14.000
1520MO	202101	GKE1	DE	EUR	30.000
1520MO	202101	GKE1	AT	EUR	200.000

Zwischenergebnis:

getDim(<IS01>,"LD")

MO	MP	EC	LD	WG	Werte
1200MO	202012	GKE1	AT	EUR	AT
3400MO	202012	GKE1	DE	EUR	DE
1520MO	202101	GKE1	DE	EUR	DE
1520MO	202101	GKE1	AT	EUR	AT

Beispiel 1:

Aufruf

aggInc(<IS01>,MO,MP,EC)

Ergebnis

LD	WG	Werte
AT	EUR	12.200.000
DE	EUR	44.000

Beispiel 2:

Aufruf

aggInc(getDim(<IS01>,"LD"), MO, MP)

Ergebnis

EC	LD	WG	Werte
GKE1	AT	EUR	AT
GKE1	DE	EUR	DE

Beispiel 3:

Aufruf

aggInc(getDim(<IS01>,"LD"), MO, MP, LD)

Ergebnis

EC	WG	Werte
GKE1	EUR	Not available

3.8.2 aggExc

Die Funktion aggExc summiert von einem Konzept alle Dimensionen, ausgenommen jener die in der Parameterliste vorkommen. In der Wertespalte werden Zahlen und Strings bei der Aggregierung berücksichtigt.

Funktionsaufruf:

aggExc(Param1,ParamN)

Beispiel:

aggExc(<IS01>, MO, MP, EC)

Parameterbeschreibung:

- Param 1 ist ein Konzept (Skalar/Vektor)
- ParamN sind 1 bis N Dimensionen welche vom Konzept unterstützt werden müssen

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number
String	String
Boolean	Boolean
Date	Date
NaN	NaN
NA	ignoreNa = True → NA wird zu NA *(wenn ausschließlich NAs vorkommen, ansonsten wird NA ignoriert)* ignoreNa = False → NA wird zu NA
#NR	#NR
Number as String	Number as String
Boolean as String	Boolean as String
Date as String	Date as String
Leere Menge	ignoreNa = True → leere Menge wird zu leere Menge ignoreNa = False → leere Menge wird zu leere Menge

Algorithmus:

Die Funktion aggExc summiert von einem Konzept alle Dimensionen, ausgenommen jene die in der Parameterliste vorkommen.

Bei einem nicht Number-Datentyp-Vektor wird ein Ergebnis geliefert, falls jede Zeile den gleichen String-Wert hat. Dann wird auch im Aggregat der gleiche Wert geliefert, sollten sich die Werte unterscheiden liefert die Funktion ein NA.

Folgende Schreibweisen sind somit äquivalent:

```
aggExc(<IS01>, MO, MP, EC) == <IS01 LD_G=ALLE/S WG_G=ALLE/S>
aggExc(<IS01>, LD, WG) == <IS01 MO_G=ALLE/S MP_G=ALLE/S EC_G=ALLE/S>
```

Beispiele:

Ausgangskonzept:

<IS01>

MO	MP	EC	LD	WG	Werte
1200MO	202012	GKE1	AT	EUR	12.000.000
3400MO	202012	GKE1	DE	EUR	14.000
1520MO	202101	GKE1	DE	EUR	30.000
1520MO	202101	GKE1	AT	EUR	200.000

Zwischenergebnis:

getDim(<IS01>,"LD")

MO	MP	EC	LD	WG	Werte
1200MO	202012	GKE1	AT	EUR	AT
3400MO	202012	GKE1	DE	EUR	DE
1520MO	202101	GKE1	DE	EUR	DE
1520MO	202101	GKE1	AT	EUR	AT

Beispiel 1:

Aufruf

aggExc(<IS01>,LD,WG)

Ergebnis

LD	WG	Werte
AT	EUR	12.200.000
DE	EUR	44.000

Beispiel 2:

Aufruf

aggExc(getDim(<IS01>,"LD"), MO, MP)

Ergebnis

MO	MP	Werte
1200MO	202012	AT
3400MO	202012	DE
1520MO	202101	NA

Beispiel 3:

Aufruf

aggExc(getDim(<IS01>,"LD"), EC, WG)

Ergebnis

EC	WG	Werte
GKE1	EUR	NA

3.8.3 sum

Funktionsbeschreibung

Berechnet die Summe des Vektors. Werden mehrere Vektoren angegeben, wird pro Zeile die Summe berechnet. Dadurch ist der Rückgabetyp ein Vektor.

Funktionsaufruf

sum(#ParamN)

Beispiel:

sum(<IS01>,<IS02>,"ignoreNa")

Parameterbeschreibung

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

Beispiele

Ausgangskonzept:

<IS01>

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

<IS02>

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	10000
5600MO	AT	1000

<IS03>

MO	LD	Werte
1200MO	AT	4000

3400MO	DE	500
5600MO	DE	0
5600MO	AT	NA

Beispiel 1

Aufruf

sum(<IS01>,<IS02>)

Ergebnis

MO	LD	Werte
1200MO	AT	6000
3400MO	DE	9000
5600MO	DE	15000
5600MO	AT	3000

Beispiel 2

Aufruf

sum(<IS01>,<IS02>,<IS03>)

Ergebnis

MO	LD	Werte
1200MO	AT	10000
3400MO	DE	9500
5600MO	DE	15000
5600MO	AT	NA

Beispiel 3

Aufruf

sum(<IS01>,<IS02>,<IS03>,"ignoreNa")

Ergebnis

MO	LD	Werte
1200MO	AT	10000
3400MO	DE	9500
5600MO	DE	15000
5600MO	AT	3000

3.8.4 min

Funktionsbeschreibung

Findet die kleinste Zahl der mitgelieferten Werte. Werden mehrere („:“) Vektoren angegeben, wird pro Zeile das Minimum gesucht. Dadurch ist der Rückgabetyp ein Vektor.

Funktionsaufruf

min(#ParamN)

Beispiel:

min(<IS01>, <IS02>, <IS03>)

Parameterbeschreibung

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

Datentypkompatibilität

Datentyp	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA
#NR	techn. Fehler

Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Beispiele

Ausgangskonzept:

<IS01>

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

<IS02>

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	NA
5600MO	AT	1000

Beispiel 1

Aufruf:

min(<IS01>)

Ergebnis:

Wert

1000

Beispiel 2

Aufruf:

min(<IS01>,<IS02>, "ignoreNA")

Ergebnis:

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	3000
5600MO	DE	5000
5600MO	AT	1000

3.8.5 max

Die Funktion max findet die größte Zahl der mitgelieferten Werte. Werden mehrere („:“) Vektoren angegeben, wird pro Zeile das Maximum gesucht. Dadurch ist der Rückgabetyp ein Vektor.

Funktionsaufruf

max(#ParamN)

Beispiel:

max(<IS01>, <IS02>, <IS03>)

Parameterbeschreibung

Parameter #ParamN ist ein IS-Konzept, OS-Konzept oder Rechenregel

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN

NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Algorithmus

Findet die größte Zahl der mitgelieferten Werte. Werden mehrere Vektoren angegeben, wird pro Zeile das Maximum gesucht. Dann ist der Rückgabetyp ein Vektor.

Beispiele

Ausgangskonzept:

<IS01>

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

<IS02>

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	NA

5600MO	AT	1000

Beispiel 1

Aufruf:

max(<IS01>)

Ergebnis:

Wert
6000

Beispiel 2

Aufruf:

max(<IS01>,<IS02>, "ignoreNA")

Ergebnis:

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

3.8.1 „/"-Funktionen

Name	Kürzel	Beschreibung
Summe	S	Summiert alle Werte
Maximum	X	Sucht den größten Wert
Minimum	I	Sucht den kleinsten Wert
Mean	M	Berechnet das arithmetische Mittel
Mean_NULL	N	Berechnet das arithmetische Mittel ohne Nullwerte
Median	Z	Berechnet den Median

Die „/“-Funktionen können mit allen Datentypen als Input arbeiten.

Bei Numeric-Vektoren werden die Werte summiert (/S), der größte Wert gesucht (/X) und die zusätzlich oben beschriebenen Funktionen durchgeführt.

Bei einem String-Datentyp-Vektor wird ein Ergebnis geliefert, falls jede Zeile den gleichen String-Wert hat. Dann wird auch im Aggregat (Maximum, Minimum, …) der gleiche Wert geliefert sollten sich die Werte unterscheiden liefert die Funktion ein NA.

Dieses Vorgehen wird auch bei den Funktionen aggExc und aggInc angewandt. Die angeführten Beispiele sollen das Vorgehen besser illustrieren.

Beispiel 1:

Annahme:

KO	FOKAC	LD	WG	Wert
IS01	A	AT	EUR	1
IS01	A	AT	USD	2
IS01	A	DE	EUR	3

Aufruf:

```
aggInc(getDim(IS01,"FOKAC"),FOKAC,WG)
```

Ergebnis:

KO	LD	Wert
IS01	AT	A
IS01	DE	A

Beispiel 2:

Annahme:

KO	FOKAC	LD	WG	Wert
IS01	A	AT	EUR	1
IS01	B	AT	USD	2
IS01	A	DE	EUR	3

Aufruf:

aggExc(getDim(IS01,"FOKAC"), LD, KO)

Ergebnis:

KO	LD	Wert
IS01	AT	NA
IS01	DE	A

Bei anderen Datentypen oder gemischten Vektor werden die Werte zu einem NA zusammengefasst, dieses Vorgehen ist oberhalb im Beispiel 2 ersichtlich.

3.9 Aggregats-Funktionen - Intervallarithmetik

3.9.1 aggInc – Intervallarithmetik

Funktionsbeschreibung

Summiert nach den Regeln der Intervallarithmetik von einem Konzept alle angegebenen Dimensionen. In der Wertespalte werden Zahlen und Strings bei der Aggregierung berücksichtigt.

Funktionsaufruf

aggInc(KO, ParamN)

Beispiel:

aggInc(<IS01>, MO, MP, EC)

Parameterbeschreibung

- KO ist ein Konzept (Skalar/Vektor)
- ParamN sind 1 bis N Dimensionen, die vom Konzept unterstützt werden müssen

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Number
String	String
Boolean	Boolean
Date	Date
NaN	NaN
NA	ignoreNa = True → NA wird zu NA *(wenn ausschließlich NAs vorkommen, ansonsten wird NA ignoriert)* ignoreNa = False → NA wird zu NA
#NR	#NR
Number as String	Number String
Boolean as String	Boolean as String
Date as String	Date as String

Leere Menge	ignoreNa = True → leere Menge wird zu Leere Menge ignoreNa = False → leere Menge wird zu Leere Menge

Algorithmus

Die Funktion aggInc nimmt ein Konzept entfernt die Dimensionen Param1 bis ParamN, wobei die Werte nach den Regeln der Intervallarithmetik summiert werden.

Bei einem nicht Number-Datentyp-Vektor wird ein Ergebnis geliefert, falls jede Zeile den gleichen String-Wert hat. Dann wird auch im Aggregat der gleiche Wert geliefert. Sollten sich die Werte unterscheiden liefert die Funktion den Wert NA.

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	WG	Werte
1200MO	AT	EUR	1000
3400MO	DE	EUR	6000
5600MO	DE	EUR	5000
5600MO	AT	EUR	2000
1200MO	AT	USD	5000
3400MO	DE	USD	3000
5600MO	DE	USD	10000
5600MO	AT	USD	1000

<IS02>

MO	LD	WG	Werte
1200MO	AT	EUR	A
3400MO	DE	EUR	B
5600MO	DE	EUR	C

5600MO	AT	EUR	D
1200MO	AT	USD	A
3400MO	DE	USD	X
5600MO	DE	USD	Y
5600MO	AT	USD	D

Beispiel 1

Aufruf

aggInc(<IS01>,LD)

Ergebnis

MO	WG	Intervall
1200MO	EUR	[500; 1500]
3400MO	EUR	[5500; 6500]
5600MO	EUR	[6000; 8000]
1200MO	USD	[4500; 5500]
3400MO	USD	[2500; 3500]
5600MO	USD	[10000; 12000]

Beispiel 2

Aufruf

aggInc(<IS01>,WG)

Ergebnis

MO	LD	Intervall
1200MO	AT	[5000; 7000]
3400MO	DE	[8000; 10000]
5600MO	DE	[14000; 16000]
5600MO	AT	[2000; 4000]

Beispiel 3

Aufruf

aggInc(<IS01>,WG, "ignoreInterval")

Ergebnis

MO	LD	Intervall
1200MO	AT	[6000; 6000]
3400MO	DE	[9000; 9000]
5600MO	DE	[15000; 15000]
5600MO	AT	[3000; 3000]

Beispiel 4

Aufruf

aggInc(<IS02>,LD, "ignoreInterval")

Ergebnis

MO	WG	Werte
1200MO	EUR	A
3400MO	EUR	B
5600MO	EUR	NA
1200MO	USD	A
3400MO	USD	X
5600MO	USD	NA

Beispiel 5

Aufruf

aggInc(<IS02>,WG, "ignoreInterval")

Ergebnis

MO	LD	Werte

1200MO	AT	A
3400MO	DE	NA
5600MO	DE	NA
5600MO	AT	D

3.9.2 *aggExc – Intervallarithmetik*

Funktionsbeschreibung

Die Funktion aggExc summiert nach den Regeln der Intervallarithmetik von einem Konzept alle Dimensionen, ausgenommen jener die in der Parameterliste vorkommen. In der Wertespalte werden Zahlen und Strings bei der Aggregierung berücksichtigt.

Funktionsaufruf

aggExc(Param1,ParamN)

Beispiel:

aggExc(<IS01>, MO, MP, EC)

Parameterbeschreibung

- Param 1 ist ein Konzept (Skalar/Vektor)
- ParamN sind 1 bis N Dimensionen welche vom Konzept unterstützt werden müssen

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Number
String	String
Boolean	Boolean
Date	Date
NaN	NaN
NA	ignoreNa = True → NA wird zu NA *(wenn ausschließlich NAs vorkommen, ansonsten wird NA ignoriert)* ignoreNa = False → NA wird zu NA
#NR	#NR

Number as String	Number String
Boolean as String	Boolean as String
Date as String	Date as String
Leere Menge	ignoreNa = True → leere Menge wird zu Leere Menge ignoreNa = False → leere Menge wird zu Leere Menge

Algorithmus

Die Funktion aggExc nimmt ein Konzept entfernt alle Dimensionen außer den Dimensionen Param1 bis ParamN, wobei die Werte nach den Regeln der Intervallarithmetik summiert werden.

Bei einem nicht Number-Datentyp-Vektor wird ein Ergebnis geliefert, falls jede Zeile den gleichen String-Wert hat. Dann wird auch im Aggregat der gleiche Wert geliefert. Sollten sich die Werte unterscheiden liefert die Funktion den Wert Na.

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	WG	Werte
1200MO	AT	EUR	1000
3400MO	DE	EUR	6000
5600MO	DE	EUR	5000
5600MO	AT	EUR	2000
1200MO	AT	USD	5000
3400MO	DE	USD	3000
5600MO	DE	USD	10000
5600MO	AT	USD	1000

<IS02>

MO	LD	WG	Werte

1200MO	AT	EUR	A
3400MO	DE	EUR	B
5600MO	DE	EUR	C
5600MO	AT	EUR	D
1200MO	AT	USD	A
3400MO	DE	USD	X
5600MO	DE	USD	Y
5600MO	AT	USD	D

Beispiel 1

Aufruf

aggExc(<IS01>,MO,WG)

Ergebnis

MO	WG	Werte
1200MO	EUR	[500; 1500]
3400MO	EUR	[5500; 6500]
5600MO	EUR	[6000; 8000]
1200MO	USD	[4500; 5500]
3400MO	USD	[2500; 3500]
5600MO	USD	[10000; 12000]

Beispiel 2

Aufruf

aggExc(<IS01>,MO,LD)

Ergebnis

MO	LD	Werte
1200MO	AT	[5000; 7000]

3400MO	DE	[8000; 10000]
5600MO	DE	[14000; 16000]
5600MO	AT	[2000; 4000]

Beispiel 3

Aufruf

aggExc(<IS01>,MO, LD, "ignoreInterval")

Ergebnis

MO	LD	Werte
1200MO	AT	[6000; 6000]
3400MO	DE	[9000; 9000]
5600MO	DE	[15000; 15000]
5600MO	AT	[3000; 3000]

Beispiel 4

Aufruf

aggExc(<IS02>,MO, WG, "ignoreInterval")

Ergebnis

MO	WG	Werte
1200MO	EUR	A
3400MO	EUR	B
5600MO	EUR	Na
1200MO	USD	A
3400MO	USD	X
5600MO	USD	Na

Beispiel 5

Aufruf

aggExc(<IS02>,MO, LD, "ignoreInterval")

Ergebnis

MO	LD	Werte
1200MO	AT	A
3400MO	DE	Na
5600MO	DE	Na
5600MO	AT	D

3.9.3 sum – Intervallarithmetik

Funktionsbeschreibung

Die Funktion sum summiert nach den Regeln der Intervallarithmetik alle n Konzept-Vektoren auf.

Funktionsaufruf

sum(#ParamN)

Beispiel:

sum(<IS01>,<IS02>,"ignoreNa")

Parameterbeschreibung

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

<IS02>, @decimals=-3

MO	LD	Werte

1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	10000
5600MO	AT	1000

<IS03>, @decimals = -3

MO	LD	Werte
1200MO	AT	4000
3400MO	DE	500
5600MO	DE	0
5600MO	AT	NA

Beispiel 1

Aufruf

sum(<IS01>,<IS02>)

Ergebnis

MO	LD	Werte
1200MO	AT	[5000; 7000]
3400MO	DE	[8000; 10000]
5600MO	DE	[14000; 16000]
5600MO	AT	[2000; 4000]

Beispiel 2

Aufruf

sum(<IS01>,<IS02>,"ignoreInterval")

Ergebnis

MO	LD	Werte
1200MO	AT	[6000; 6000]

3400MO	DE	[9000; 9000]
5600MO	DE	[15000; 15000]
5600MO	AT	[3000; 3000]

Beispiel 3

Aufruf

sum(<IS01>,<IS02>,<IS03>)

Ergebnis

MO	LD	Werte
1200MO	AT	[8500; 11500]
3400MO	DE	[8000; 11000]
5600MO	DE	[13500; 16500]
5600MO	AT	NA

Beispiel 4

Aufruf

sum(<IS01>,<IS02>,<IS03>,"ignoreInterval")

Ergebnis

MO	LD	Werte
1200MO	AT	[10000; 10000]
3400MO	DE	[9500; 9500]
5600MO	DE	[15000; 15000]
5600MO	AT	NA

Beispiel 5

Aufruf

sum(<IS01>,<IS02>,<IS03>,"ignoreInterval","ignoreNa")

Ergebnis

MO	LD	Werte
1200MO	AT	[10000; 10000]
3400MO	DE	[9500; 9500]
5600MO	DE	[15000; 15000]
5600MO	AT	[3000; 3000]

Beispiel 6

Aufruf

sum(<IS01>,<IS02>,<IS03>,"ignoreNa")

Ergebnis

MO	LD	Werte
1200MO	AT	[8500; 11500]
3400MO	DE	[8000; 11000]
5600MO	DE	[13500; 16500]
5600MO	AT	[2000; 4000]

3.9.4 min – Intervallarithmetik

Funktionsbeschreibung

Findet die kleinste Zahl der mitgelieferten Werte. Werden mehrere Vektoren angegeben, wird pro Zeile das Minimum gesucht. Dann ist der Rückgabetyp ein Vektor.

Funktionsaufruf

min(#ParamN)

Beispiel:

min(<IS01>, <IS02>, <IS03>)

Parameterbeschreibung

Parameter #ParamN ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert.

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis

Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

<IS02>, @decimals=-3

MO	LD	Werte

1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	NA
5600MO	AT	1000

Beispiel 1

Aufruf:

min(<IS01>)

Ergebnis:

Wert
[500; 1500]

Beispiel 2

Aufruf:

min(<IS01>,"ignoreInterval")

Ergebnis:

Wert
[1000; 1000]

Beispiel 3

Aufruf:

min(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	[500; 1500]
3400MO	DE	[2500; 3500]
5600MO	DE	NA
5600MO	AT	[500; 1500]

Beispiel 4

Aufruf:

min(<IS01>,<IS02>, "ignoreInterval","ignoreNA")

Ergebnis:

MO	LD	Werte
1200MO	AT	[1000; 1000]
3400MO	DE	[3000; 3000]
5600MO	DE	[5000; 5000]
5600MO	AT	[1000; 1000]

3.9.5 max – Intervallarithmetik

Funktionsbeschreibung

Findet die größte Zahl der mitgelieferten Werte. Werden mehrere Vektoren angegeben, wird pro Zeile das Maximum gesucht. Dann ist der Rückgabetyp ein Vektor.

Funktionsaufruf

max(#ParamN)

Beispiel:

max(<IS01>, <IS02>, <IS03>)

Parameterbeschreibung

- #ParamN sind 1 bis n IS-Konzepte, OS-Konzepte oder Rechenregeln

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA

#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	5000
5600MO	AT	2000

<IS02>, @decimals=-3

MO	LD	Werte
1200MO	AT	5000
3400MO	DE	3000
5600MO	DE	NA
5600MO	AT	1000

Beispiel 1

Aufruf:

max(<IS01>)

Ergebnis:

Wert
[5500; 6500]

Beispiel 2

Aufruf:

max(<IS01>,"ignoreInterval")

Ergebnis:

Wert
[6000; 6000]

Beispiel 3

Aufruf:

max(<IS01>,<IS02>)

Ergebnis:

MO	LD	Werte
1200MO	AT	[4500; 5500]
3400MO	DE	[5500; 6500]
5600MO	DE	NA
5600MO	AT	[1500; 2500]

Beispiel 4

Aufruf:

max(<IS01>,<IS02>, "ignoreInterval","ignoreNA")

Ergebnis:

MO	LD	Werte
1200MO	AT	[5000; 5000]
3400MO	DE	[6000; 6000]
5600MO	DE	[5000; 5000]
5600MO	AT	[2000; 2000]

3.9.1 „/"-Funktionen – Intervallarithmetik

Name	Kürzel	Beschreibung
Summe	S	Summiert alle Werte
Maximum	X	Sucht den größten Wert
Minimum	I	Sucht den kleinsten Wert

Die „/"-Funktionen können mit allen Datentypen als Input arbeiten.

Bei Numeric-Vektoren werden die Werte summiert (/S), der größte Wert gesucht (/X) und die zusätzlich oben beschriebenen Funktionen durchgeführt.

Bei einem String-Datentyp-Vektor wird ein Ergebnis geliefert, falls jede Zeile den gleichen String-Wert hat. Dann wird auch im Aggregat (Maximum, Minimum, …) der gleiche Wert geliefert sollten sich die Werte unterscheiden liefert die Funktion ein Na.

Dieses Vorgehen wird auch bei den Funktionen aggExc und aggInc angewandt. Dies Verhält sich ident zu Sektion 3.8.1 in der Punkt-Arithmetik. Der Unterschied zur Punkt-Arithmetik liegt jedoch darin, dass die „/"-Funktionen hier nur für Summe, sowie Minimum und Maximum angewandt werden können.

3.10 Stammdaten-Funktionen

Alle Stammdatenfunktionen werden in Prüfungen verwendet.

3.10.1 getAttribute

Über die getAttribute-Methode kann man aus definierten Stammdatentabellen (z.B. WP, Ident) einzelne Stammdatenwerte auslesen. So kann man z.B. aus der WP-Stammtabelle das Nominale auslesen.

Funktionsaufruf:

getAttribute(Konzept, Dimension, AttributeZurDimension,“Konzeptwert“, „Referenzperiode“)

Beispiel:

getAttribute(<ISWPSCW01_WP THVKZ=F WA=EM>, "WK", "EMISSIONS_VOLUMEN", MP=+1M)

Parameterbeschreibung:

1. Parameter: z.B. <ISWPSCW01_WP THVKZ=F WA=EM>

   Im ersten Parameter müssen Dimensionswerte geliefert werden, die einen Satz aus der jeweiligen Stammdatentabelle eindeutig referenzieren. Im Falle der Wertpapierstammdaten wäre das die ISIN. Es muss in diesem Falle also entweder eine ISIN direkt angegeben werden z.B. AT0000652011 (Erste Bank Group) oder wie im obigen Beispiel ein Konzept, das als Dimension die ISIN hat. In Fall, dass ein Konzept angegeben wird, muss hinterlegt werden, wie die identifizierende Dimension heißt. Hier wäre das die ISIN, die sich in der Dimension SC_ISIN (Code WK) befindet. Diese Information muss beim Mapping von WP_STAMM hinterlegt werden.

2. Parameter: z.B. "WK"

   Der Dataprovider liefert eine Liste an Werten (mit Dimensionen) mit der Einschränkung TKVKZ=F und WA=EM und die Funktion getAttribute nimmt sich aus diesen Werten alle Dimensionswerte der Dimension WK, was eine Liste an ISINs ist.

3. Parameter: z.B. "EMISSIONS_VOLUMEN"

   Mit dem Parameter wird die Spalte angegeben, deren Wert als Ergebnis aus dem OeNB internen System zurückgeliefert werden soll. Im Falle des Beispiels wäre es die Spalte EMISSIONS_VOLUMEN. Bei der Dimension MO (Meldeobjekt) wird zusätzlich ein Präfix MO_ mitgeführt. Als Beispiel MO_ FIRMENBUCH_NR liefert die Firmenbuch-Nummer des Melders. Die Dimension MO bezieht sich auf die „melder_id“ des Melder-XMLs.

4. Parameter: "Konzeptwert"

   Dieser Parameter ist optional und regelt, ob ein Dimensionswert oder der Konzeptwert für die Stammdatenabfrage verwendet wird.

5. Parameter: "Referenzperiode“

Mit dem optionalen Parameter "Referenzperiode" werden die Stammdaten aus der jeweiligen Referenzperiode rückgeliefert. Es können relative (z.B. MP=+1M) oder absolute Werte (z.B. MP=202012) angegeben werden.

3.10.2 getBez

Die Funktion schreibt zu einer Dimension eines Konzepts alle zugehörigen Werte einer bestimmten Beziehungsart in eine neue zusätzliche Dimension.

Funktionsaufruf:

getBez(Konzept, Dimensionscode, Objektart, Richtung, "Bezeichnung der Ergebnisdimension", [Filterspalte, Filterwert])

Parameterbeschreibung:

1. Parameter Konzept
- ist ein IS-Konzept oder OS-Konzept
2. Parameter Dimensionscode
- ist eine Dimension vom Konzept
3. Parameter Objektart
- ist die Beziehungsart (Objektart)
4. Parameter Richtung
- ist "BEZ_AKTIV" oder "BEZ_PASSIV", gibt die abzufragende Richtung der Beziehung an.
5. Parameter "Bezeichnung der Ergebnisdimension"
- ist die Bezeichnung der neu hinzugefügten Dimension (z.B. "INC" oder "LD" – Muss eine vorhandene Dimension sein)
6. Parameter Filterspalte
- ist optional und soll als Filter für Beziehungsattribute dienen
7. Parameter Filterwert
- ist optional und soll als Filter für Beziehungsattribute dienen
-

Beispiel 1:

Annahme:

KO	MO	MP	IDUNT (= Ident-Nr.)	WA	Wert
IS01	30937	31.01.2017	30937	BW	100
IS01	30937	31.01.2017	11342854	BW	100
IS01	30937	31.01.2017	8382948	BW	100
IS01	30000	31.01.2017	30000	MW	200
IS01	40000	31.01.2017	40000	NN	100
IS01	50000	31.01.2017	50000	WS	100

Erklärung:

MO 30937 meldet für sich selbst (es folgt: Zeile mit IDUNT 30937 und für die Zweigniederlassung 11342854 und 8382948)

MO 30000, 40000, 50000 melden nur für sich selbst und haben daher bei IDUNT dasselbe wie bei MO stehen. Zu Meldeobjekt/Ident-Nr. 30000, 40000, 50000 sind keine Zweigniederlassungen definiert

Aufruf:

getBez(IS01, IDUNT, znbez, BEZ_PASSIV, "Hauptanstalt")

Erklärung:

IS01 ist das Konzept und IDUNT ist der Dimensionscode, znbez ist die Objektart, BEZ_PASSIV ist die Richtung, "Hauptanstalt" die Bezeichnung der neuen Dimension

Zu IdentNr 11342854 wird die Hauptniederlassung IdentNr 30937 geliefert und in die neue Dimension "Hauptanstalt" geschrieben.

Bei gemeldeten Zeilen mit IDUNT 30937 ist dies bereits die Hauptniederlassung und es wird in die neue Dimension "Hauptanstalt" der Wert #NR geschrieben.

Ergebnis:

KO	MO	MP	IDUNT (= Ident-Nr.)	WA	Hauptanstalt	Wert
IS01	30937	31.01.2017	30937	BW	#NR	100
IS01	30937	31.01.2017	11342854	BW	30937	100
IS01	30937	31.01.2017	8382948	BW	30937	100
IS01	30000	31.01.2017	30000	MW	#NR	200
IS01	40000	31.01.2017	40000	NN	#NR	100
IS01	50000	31.01.2017	50000	WS	#NR	100

3.10.3 getBezAttribute

Über die getBezAttribute-Methode kann man aus definierten Stammdatentabellen (z.B. Ident) einzelne Beziehungsattribute auslesen. So kann man z.B. aus der Ident-Nummer-Stammtabelle den Anteil in EUR auslesen.

Bei dieser Funktion wird #NR wie NA interpretiert.

Funktionsaufruf:

getBezAttribute(Konzept, aktivIdent, passivIdent, ignoriereRichtung, Objektart, Beziehungsattribut, [Filterspalte, Filterwert])

Parameterbeschreibung:

1. Parameter Konzept
- ist ein IS-Konzept oder OS-Konzept
2. Parameter aktivIdent
- ist die aktive Seite der Beziehung
3. Parameter passivIdent
- ist die passive Seite der Beziehung
4. Parameter ignoriereRichtung
- ist entweder „True“, falls die aktiv und passiv Seite für das Attribut nicht wichtig ist.
5. Parameter Objektart
- ist die Beziehungsart für welche das Beziehungsattribut ausgewählt wird z.B. "DI-Gesellschafter"
6. Parameter Beziehungsattribut
- ist das Beziehungsattribut, welches abgefragt wird
7. Parameter Filterspalte
- ist optional und sollen als Filter für Beziehungsattribute dienen
8. Parameter Filterwert
- ist optional und sollen als Filter für Beziehungsattribute dienen
-

Beispiel 1:

Annahme:

Stammdaten

IDENT_NR_AKTIV	IDENT_NR_PASSIV	OBJ_ART	WIRKSAM_VON	WIRKSAM_BIS	WF_STATUS	ANTEIL_PROZENT	ANTEIL_BETRAG
53767	3537021	digesellbez	31.12.2007	01.10.2008	gueltig	100	64169000
53767	6922309	digesellbez	31.12.2006		gueltig	100	1000

KO	MP	MO	IDAKT	IDPAS	Wert
OS1234	30.07.2008	53767	53767	3537021	0
OS1234	31.10.2017	53767	53767	6922309	123

Aufruf:

getBezAttribute(<OS1234>, "IDAKT", "IDPAS", False, "ANTEIL_BETRAG_WHG", "digesellbez")

Ergebnis:

MP	MO	IDAKT	IDPAS	Wert
30.07.2008	53767	53767	3537021	64169000
31.10.2017	53767	53767	6922309	1000

3.10.4 getHoechste

Die Funktion getHoechste() soll gemäß der (explizit oder über OS/RR) absteigend angegebenen Institutsart-Hierarchie die höchste beim angegebenen Konzept vorliegende ILZ ermitteln, das angegebene Konzept (oder die optional angegebenen ANUBIS-Formel) mit dieser ILZ auswerten und die Dimension MO mit dieser höchsten ILZ befüllen. Wenn der optionale Parameter „moToBlz“ gesetzt ist, wird im Ergebnisvektor in der Dimension „MO“ die BLZ des MOs zurückgegeben um ILZ-übergreifende Berechnungen zu ermöglichen.

Funktionsaufruf:

getHoechste(#Param1, Param#2, #Param3, #Param4)

Parameterbeschreibung:

1. Parameter #Param1
   ist ein IS-Konzept, OS-Konzept oder eine RR, anhand dessen die höchste Institutsart gefunden werden soll.
2. Parameter #Param2
   ist eine Hierarchie an Institutsarten (in " " und getrennt durch ","). Statt direkt die Hierarchie anzuführen kann auch auf eine RR / ein OS-Konzept verwiesen werden in der die Hierarchie als String in der gleichen Syntax angeführt wird.
3. Parameter #Param3 (optional)
   ist eine ANUBIS-Funktion, RR oder OS die mit der gefundenen höchsten Institutsart durchgeführt werden soll.
4. Parameter #Param4 (optional)
   ist ein Fixwert „moToBlz“

Beispiel:

getHoechste(IS01,"VB,F1,FH,KA,KK,KI","moToBlz")

getHoechste(IS01,"VB,F1,FH,KA,KK,KI",IS01+IS02+IS03 ,"moToBlz")

Datentypkompatibilität:

Datentyp (Eingabe)	Ergebnis
Number	Ergebnis lt. Syntaxberechnung
String	Ergebnis lt. Syntaxberechnung
Boolean	Ergebnis lt. Syntaxberechnung

Date	Ergebnis lt. Syntaxberechnung
NaN	NaN
NA	ignoreNa = True > NA wird im 3. Param. nach NA Logik ersetzt ignoreNa = False → NA wird zu NA im 3. Param.
#NR	#NR
Number as String	Ergebnis lt. Syntaxberechnung
Boolean as String	Ergebnis lt. Syntaxberechnung
Date as String	Ergebnis lt. Syntaxberechnung
Leere Menge	ignoreNa = True > NA wird im 3. Param. nach NA Logik ersetzt ignoreNa = False → NA wird zu NA im 3. Param.

Beispiel 1:

Annahme:

MP	KO	MO	EC	Wert
201712	ISFIN1000000	12000FH	FINKD	3
201712	ISFIN1000000	12000KA	51	5
201712	ISFIN1000000	1200MO	51	2
201712	ISFIN1000000	12000TK	51	4

Aufruf:

getHoechste(<ISFIN1000000 EC=51>,"VB,F1,FH,KA,KK,KI")

Ergebnis:

MP	KO	MO	EC	Wert
201712	ISFIN1000000	12000KA	51	5

Beispiel 2:

Annahme:

MP	KO	MO	EC	Wert

201712	ISFIN1000000	12000FH	FINKD	3
201712	ISFIN1000000	12000KA	51	5
201712	ISFIN1000000	12000FH	51	4
201712	ISFIN1000000	12000TK	51	4
201712	ISCAT7980000	12000FH	81	1

Aufruf:

getHoechste(<ISFIN1000000 EC=51>,RR123, RRABC)
RR123 = "VB,F1,FH,KA,KK,KI"
RRABC = <ISCAT7980000 EC_G=ALLE/X>/<ISFIN1000000 EC_G=ALLE/X>

Ergebnis:

MP	KO	MO	EC	Wert
201712	ISFIN1000000	12000FH	#NR	0,25

3.10.5 getMeldepflicht

Die Funktion getMeldePflicht liefert die Meldepflicht zu einem Erhebungscode in der Form TRUE/FALSE (ist nicht in der Datenbank vorhanden) für alle meldepflichtigen Institute zurück (analog zum Konzept ISPOSQ). Der Rückgabevektor hat die Dimensionen MO, MP, EC.

Funktionsaufruf:

getMeldePflicht(#Erhebung)

Parameterbeschreibung

1. Parameter #Erhebung (optional)
   ist ein Erhebungscode als Text, zB. „80“

3.10.6 getMeldeTermin

Die Funktion getMeldetermin liefert den Meldetermin in Form von YYYYMMDDHHMMSS als BigDemical (Skalar ohne Dimensionen) zurück (analog zum Konzept ISPOSQ).

Funktionsaufruf:

getMeldetermin(#Erhebung)

Parameterbeschreibung:

1. Parameter #Erhebung (optional)
   ist ein Erhebungscode als Text, zB. „80“

3.10.7 getNeuinsolvenz

Die Funktion getNeuinsolvenz liefert einen Vektor mit Neuinsolvenzen für den angegebenen Referenzzeitraum. Der Vektor hat immer 5 Dimensionen (IN, INSSIGEL, GLTGVON, INSSIGELNR, INSTX). Es werden prinzipiell alle Ergebnisse zurückgeliefert bei denen kein Gültig-bis gesetzt ist und wo der Datensatz im Status "aktiv" ist. Relative Zeitangaben erfolgen analog zu den Zeitangaben der Meldeperiode (+/- nT, nM, nQ, nJ), jedoch auch mit positivem Vorzeichen. Sie beziehen sich immer relativ auf die Aktualisierungsperiode.

Funktionsaufruf:

getNeuinsolvenz(Para1, Para2, [Para3])

Parameterbeschreibung:

1. Parameter Para1
   ist der Bezugspunkt „von“ des Referenzzeitraums und kann als relativer Zeitraum oder als Datum angegeben werden, zB. 20181231, -1J. Der Parameter kann auch ein Konzept sein, wenn das Konzept einen Timestamp-Skalar enthält.
2. Parameter Para2
   ist der Bezugspunkt „bis“ des Referenzzeitraums und kann als relativer Zeitraum oder als Datum angegeben werden, zB. 20181231, -1J. Der Parameter kann auch ein Konzept sein, wenn das Konzept einen Timestamp-Skalar enthält.
3. Parameter Param3 (optional)
   ist ein Filterparameter der angibt auf welche Sigel gefiltert werden soll. Elemente werden mit Beistrichen getrennt in eckigen Klammern angegeben, zB. [EK, ES, NI, ZI, VO, VK].

Beispiele:

getNeuinsolvenz(20181231,20191231)
→ liefert alle Neuinsolvenzen die zwischen 31.12.2018 und 31.12.2019 an OSIRIS geliefert wurden

getNeuinsolvenz(20181231,+1J)
→ liefert alle Neuinsolvenzen die zwischen 31.12.2018 und 31.12.2019 an OSIRIS geliefert wurden

getNeuinsolvenz(20181231,+1J, "[EK, ES, NI, ZI, VO, VK]")
→ liefert alle Neuinsolvenzen die zwischen 31.12.2018 und 31.12.2019 an OSIRIS geliefert wurden mit den Sigeln EK, ES, NI, ZI, VO und VK

3.10.8 crossJoin

Bei diesem Join werden die Dimensionen von zwei Konzepten aufgefüllt. Der crossJoin verhält sich ähnlich zu einem Full Outer Join, wobei nicht vorhandene Dimensionen mit der Ausprägung #NR aufgefüllt werden.

Funktionsaufruf:

crossJoin(Konzept1, Konzept2, "[Dimliste]", SplitDim, SplitValueFrom, SplitValueTo, wayOfSplit, includeTotalAmount)

Parameterbeschreibung:

1. Parameter Konzept1

   ist ein IS-Konzept oder OS-Konzept

2. Parameter Konzept2

   ist ein IS-Konzept oder OS-Konzept

3. Parameter "[Dimliste]"

- wird als Liste angegeben z.B. [LD, WG]

Optional können nun noch zusätzlich folgende Parameter angegeben werden:

4. Parameter SplitDim

- Dimension anhand der aufgeteilt werden soll

5. Parameter SplitValueFrom

   Wertausprägung ausgehend von der aufgespalten wird (z.B.: WS)

6. Parameter SplitValueTo

   Wertausprägung auf die aufgespalten wird (z.B.: BW)

7. Parameter wayOfSplit

   Art der Aufspaltung (EQUALLY, ALIQUOT)

8. Parameter includeTotalAmount

   „includeTotalAmount“ Dimension Aufteilungsart (AFTAR) wird hinzugefügt mit folgenden Wertausprägungen (gleichmäßiger Anteil, aliquoter Anteil, Gesamtbetrag) (siehe Beispiele), wobei dieser Parameter wiederum optional ist

Beispiel 1:

Annahme:

IS01 besteht aus MO, MP, IN, INC, LD und hat oBdA nur eine Zeile

MO	MP	IN	INC	LD	Wert
1400MO	31.12.2016	388815	01	AT	5

IS02 besteht aus MO, MP, IN, INC, WG und hat oBdA nur eine Zeile

MO	MP	IN	INC	WG	Wert
1400MO	31.12.2016	388815	02	EUR	10

Der Funktonsaufruf wäre crossJoin(IS01, IS02, "[MO,MP,IN]").

Das Ergebnis soll folgendermaßen aussehen:

Im ersten Schritt werden die Konzepte zusammengehängt und die Spalten vergrößert

MO	MP	IN	INC	LD	WG	Wert
1400MO	31.12.2016	388815	01	AT		5
1400MO	31.12.2016	388815	02		EUR	10

Im zweiten Schritt werden die Dimensionen erweitert.

MO	MP	IN	INC	LD	WG	Wert
1400MO	31.12.2016	388815	01	AT	EUR	5
1400MO	31.12.2016	388815	02	AT	EUR	10

Sonderfall:

Falls beide Konzepte nur eine Zeile haben wird der Wert des ersten Konzepts genommen.

IS01 besteht aus MO, MP, IN, INC, LD und hat nur eine Zeile

MO	MP	IN	INC	LD	Wert
1400MO	31.12.2016	388815	01	AT	5

IS02 besteht aus MO, MP, IN, INC und hat nur eine Zeile

MO	MP	IN	INC	Wert

1400MO	31.12.2016	388815	02	10

Der Funktonsaufruf wäre crossJoin(IS01, IS02, "[MO,MP,IN]").

MO	MP	IN	INC	LD	Wert
1400MO	31.12.2016	388815	01	AT	5

Beispiel 2:

Keine eindeutigen Dimensionen:

Sollten die restlichen Dimensionen (ohne der DimListe) nicht eindeutig sein, ist kein Fehler auszugeben. Es sind die Zeilen zu verdoppeln.

Annahme:

IS01:

MO	MP	IN	LD	WA	Wert
1400MO	31.12.2016	388815	AT	BW	5
1400MO	31.12.2016	388815	DE	MW	20

IS02:

MO	MP	IN	WG	WA	Wert
1400MO	31.12.2016	388815	EUR	ZS	10

WICHTIG: WA muss überall unterschiedlich sein sonst kann es nicht weggespeichert werden!!!

crossJoin(IS01, IS02, "[MO,MP,IN]").

MO	MP	IN	LD	WG	WA	Wert
1400MO	31.12.2016	388815	AT	EUR	BW	5
1400MO	31.12.2016	388815	DE	EUR	MW	20
1400MO	31.12.2016	388815	?	EUR	ZS	10

Wegen der nicht Eindeutigkeit von LD von IS01 kann LD von IS02 nicht aufgefüllt werden. Somit sollen die Zeilen verdoppelt werden und aufgefüllt werden.

MO	MP	IN	LD	WG	WA	Wert
1400MO	31.12.2016	388815	AT	EUR	BW	5
1400MO	31.12.2016	388815	DE	EUR	MW	20
1400MO	31.12.2016	388815	AT	EUR	ZS	10
1400MO	31.12.2016	388815	DE	EUR	ZS	10

Weitere Beispiele:

Ausgangskonzept IS01 und IS02 werden gejoined. Dim1, Dim2 und Dim3 sollen eine beliebige Dimension (z.B. LD) symbolisieren und nur der Veranschaulichung des Joins dienen.

Ausgangskonzepte								
08. Daten zu Instrument-empfangene Sicherheit								
KO	Instrument ID	SicherheitenID	MO	MP	Dim1	Dim2	WA	Wert
IS01	1	3	12000	31.01.2017	a	b	BW	100
IS01	2	3	12000	31.01.2017	a	b	BW	200
IS01	3	3	12000	31.01.2017	a	b	BW	100
IS01	4	3	12000	31.01.2017	a	b	BW	100
07. Daten zu empfangenen Sicherheiten								
KO	SicherheitenID	MO	MP	Dim3		WA	Wert	
IS02	3	12000	31.01.2017	C		WS	1000	

Beispiel 3:

Aufruf:

crossJoin(IS01, IS02, "[SicherheitenID, MO, MP]", WA, WS, BW, equally)

Resultat 1: gleichmäßig verteilen		für Sicherheiten ID (3) soll WA 'WS' auf die Wertart 'BW' gleichmäßig aufgeteilt werden (1000 / 4 = 250)							
Neues Konzept									
KO	Instrument ID	SicherheitenID	MO	MP	Dim1	Dim2	Dim3	WA	Wert
IS03	1	3	12000	31.01.2017	a	b	c	BW	100
IS03	1	3	12000	31.01.2017	a	b	c	WS	250
IS03	2	3	12000	31.01.2017	a	b	c	BW	200
IS03	2	3	12000	31.01.2017	a	b	c	WS	250
IS03	3	3	12000	31.01.2017	a	b	c	BW	100
IS03	3	3	12000	31.01.2017	a	b	c	WS	250
IS03	4	3	12000	31.01.2017	a	b	c	BW	100
IS03	4	3	12000	31.01.2017	a	b	c	WS	250

Beispiel 4:

Aufruf:

crossJoin(IS01, IS02, "[SicherheitenID, MO, MP]", WA, WS, BW, ALIQUOT)

Resultat 2: Verteilung Aliquot		für Sicherheiten ID (3) soll WA 'WS' auf die Wertart 'BW' aliquot aufgeteilt werden (z.B. 1000/(100+200+100+100)*100 = 200)							
Neues Konzept									
KO	Instrument ID	Sicherheiten ID	MO	MP	Dim1	Dim2	Dim3	WA	Wert
IS03	1	3	12000	31.01.2017	a	b	c	BW	100
IS03	1	3	12000	31.01.2017	a	b	c	WS	**200**
IS03	2	3	12000	31.01.2017	a	b	c	BW	200
IS03	2	3	12000	31.01.2017	a	b	c	WS	**400**
IS03	3	3	12000	31.01.2017	a	b	c	BW	100
IS03	3	3	12000	31.01.2017	a	b	c	WS	**200**

IS03	4	3	12000	31.01.2017	a	b	c	BW	100
IS03	4	3	12000	31.01.2017	a	b	c	WS	200

Beispiel 5:

Aufruf:

crossJoin(IS01, IS02, "[SicherheitenID, MO, MP]", WA, WS, BW, EQUALLY, „includeTotalAmount“)

Resultat 3: Verteilung Gleichmäßig inkl. Gesamtbetrag	für Sicherheiten ID (3) soll WA 'WS' auf die Wertart 'BW' gleichmäßig aufgeteilt werden und die Dimension Aufteilungsart wird eingefügt und mit der entsprechenden Wertausprägung befüllt.

Neues Konzept										
KO	Instr. ID	Sicherh. ID	MO	MP	Dim1	Dim2	Dim3	Aufteilungsart (AFTAR)	WA	Wert
IS03	1	3	12000	31.01.2017	a	b	c	#NR	BW	100
IS03	1	3	12000	31.01.2017	a	b	c	gleichmäßiger Anteil	WS	250
IS03	1	3	12000	31.01.2017	a	b	c	Gesamtbetrag	WS	1000
IS03	2	3	12000	31.01.2017	a	b	c	#NR	BW	200
IS03	2	3	12000	31.01.2017	a	b	c	gleichmäßiger Anteil	WS	250
IS03	2	3	12000	31.01.2017	a	b	c	Gesamtbetrag	WS	1000
IS03	3	3	12000	31.01.2017	a	b	c	#NR	BW	100
IS03	3	3	12000	31.01.2017	a	b	c	gleichmäßiger Anteil	WS	250
IS03	3	3	12000	31.01.2017	a	b	c	Gesamtbetrag	WS	1000
IS03	4	3	12000	31.01.2017	a	b	c	#NR	BW	100
IS03	4	3	12000	31.01.2017	a	b	c	gleichmäßiger Anteil	WS	250
IS03	4	3	12000	31.01.2017	a	b	c	Gesamtbetrag	WS	1000

Beispiel 6:

Aufruf:

crossJoin(IS01, IS02, "[SicherheitenID, MO, MP]", WA, WS, BW, ALIQUOT,“includeTotalAmount“)

Resultat 4: Verteilung Aliquot inkl. Gesamtbetrag			für Sicherheiten ID (3) soll WA 'WS' auf die Wertart 'BW' aliquot aufgeteilt werden und die Dimension Aufteilungsart wird eingefügt und mit der entsprechenden Wertausprägung befüllt.							
Neues Konzept										
KO	Instrument ID	Sicherheiten ID	MO	MP	Dim1	Dim2	Dim3	Aufteilungsart (AFTAR)	WA	Wert
IS03	1	3	12000	31.01.2017	a	b	c	#NR	BW	100
IS03	1	3	12000	31.01.2017	a	b	c	aliquoter Anteil	WS	200
IS03	1	3	12000	31.01.2017	a	b	c	Gesamtbetrag	WS	1000
IS03	2	3	12000	31.01.2017	a	b	c	#NR	BW	200
IS03	2	3	12000	31.01.2017	a	b	c	aliquoter Anteil	WS	400
IS03	2	3	12000	31.01.2017	a	b	c	Gesamtbetrag	WS	1000
IS03	3	3	12000	31.01.2017	a	b	c	#NR	BW	100
IS03	3	3	12000	31.01.2017	a	b	c	aliquoter Anteil	WS	200
IS03	3	3	12000	31.01.2017	a	b	c	Gesamtbetrag	WS	1000
IS03	4	3	12000	31.01.2017	a	b	c	#NR	BW	100
IS03	4	3	12000	31.01.2017	a	b	c	aliquoter Anteil	WS	200
IS03	4	3	12000	31.01.2017	a	b	c	Gesamtbetrag	WS	1000

3.10.9 isMember

Die Funktion isMember stellt fest, ob die Dimensionswerte eines Konzeptes in einer vordefinierten Gruppe (Sektor/Menge) vorhanden sind oder nicht.

Funktionsaufruf:

isMember(Vektor, Sektor, Dimension, "Konzeptwert")

Parameterbeschreibung:

1. Parameter Vektor

   ist ein IS-Konzept, OS-Konzept, Rechenregel oder eine Prüfregel-Syntax die einen Vektor zurückgibt.

2. Parameter Sektor

   ist eine vordefinierte Gruppe (Menge, Sektor) von Dimensionsausprägungen der Dimension (des Vektors,). Für weitere Informationen zu Sektoren/Gruppen/Mengen siehe Kapitel 2.3.1.

3. Parameter Dimension

- Ist eine Dimension des IS-Konzept, OS-Konzept, Rechenregel oder eine Prüfregel-Syntax

4. Parameter "Konzeptwert"

   Dieser Parameter ist optional und regelt, ob ein Dimensionswert oder der Konzeptwert für die Stammdatenabfrage verwendet wird.

Beispiel 1:

Annahme:

Konzept	LD	LDC	Wert
IS01123	AT	AT	0
IS01123	DE	DE	1
IS01123	PL	US	1
IS01123	US	CU	2
IS01123	AT	PL	3
IS01123	CU	US	5
IS01123	KW	KW	8

BSATPL besteht aus den Ländern Österreich (AT) und Polen (PL)

Aufruf:

isMember (<IS01123>, “BSATPL”, “LD”)

Ergebnis:

Konzept	LD	LDC	Wert
IS01123	AT	AT	TRUE
IS01123	DE	DE	FALSE
IS01123	PL	US	TRUE
IS01123	US	CU	FALSE
IS01123	AT	PL	TRUE
IS01123	CU	US	FALSE
IS01123	KW	KW	FALSE

Beispiel 2:

Annahme:

Konzept	LD	Wert
IS01123	DE	AT
IS01123	AT	DE
IS01123	KW	PL
IS01123	US	CU
IS01123	CU	US
IS01123	PL	KW

BSATPL besteht aus den Ländern Österreich (AT) und Polen (PL)

Aufruf:

isMember (<IS01123>, “BSATPL”, “LD”, "Konzeptwert")

Ergebnis:

Konzept	LD	Wert
IS01123	DE	TRUE
IS01123	AT	FALSE
IS01123	KW	TRUE
IS01123	US	FALSE
IS01123	CU	FALSE
IS01123	PL	FALSE

3.10.10 pruefFremdschluessel

Mit Hilfe der pruefFremdschluessel-Methode kann für Identnummern bestimmt werden, ob es im Stammdatensystem der OeNB einen Fremdschlüssel gibt. Für Österreich gelten unter anderem (Stichtag 11.07.2019) folgende Codes als Fremdschlüssel.

AT_FB_CD	Firmenbuchnummer
AT_GEM_CD	Gemeindenummer
AT_IDENT_CD	Identnummer
AT_LAE_CD	Ländernummer
AT_ZVR_CD	Vereinsregisternummer

Funktionsaufruf:

pruefFremdschluessel(Konzept, KonzeptDimension, Kategorie, Filter)

Parameterbeschreibung:

1. Parameter Konzept
- ist ein IS-Konzept oder OS-Konzept
2. Parameter KonzeptDimension
- ist die eine Dimension vom Konzept,
3. Parameter Kategorie
- ist optional und gibt an, dass danach eine Filterliste anzugeben ist
4. Parameter Filter
- ist ein optionaler Parameter, falls es Einschränkungen auf bestimmte Fremdschlüsseln gibt (z.B. RIAD-Code)

Beispiel 1:

Annahme:

MP	MO	KO	SCHID	SCHLD	Wert
201712	20221MO	ISBSPCWERT01	5586577	DE	22.887,68

OeNB-Stammdaten zur SCHID 5586577:

SCHID	KATEGORIE	SCHLUESSEL	AMTSGERICHT
5586577	RIAD Code	DE05740	
5586577	DE_HRA_CD	HRA15277	R1101

Aufruf:

pruefFremdschluessel(<ISBSPCWERT01 SCHID=5586577 SCHLD=DE>,
SCHID,KATEGORIE,"DE_HRA_CD")

Ergebnis:

MP	MO	KO	SCHID	SCHLD	Wert
201712	20221MO	ISBSPCWERT01	5586577	DE	True

3.10.11 pruefWKN

Die Funktion pruefWKN prüft ob der Inhalt einer Dimension einem zu diesem Stichtag gültigen Wertpapier in OBServ (bei Verwendung in ISIS) bzw. OSIRIS entspricht. Als Ergebnis wird TRUE/FALSE geliefert.

Funktionsaufruf:

pruefWKN(#Param1, #Param2)

Parameterbeschreibung:

1. Parameter #Param1
   ist ein IS-Konzept, OS-Konzept, Rechenregel oder Syntax
2. Parameter #Param2
   ist eine Dimension des #Param2

Datentypkompatibilität:

Datentyp (Eingabe)	Ergebnis
Number	TRUE/FALSE
String	TRUE/FALSE
Boolean	TRUE/FALSE
Date	TRUE/FALSE
NaN	NaN
NA	ignoreNa = True → NA wird zu TRUE/FALSE ignoreNa = False → NA wird zu NA
#NR	TRUE/FALSE
Number as String	TRUE/FALSE
Boolean as String	TRUE/FALSE
Date as String	TRUE/FALSE
Leere Menge	ignoreNa = True → leere Menge wird zu TRUE/FALSE ignoreNa = False → leere Menge wird zu NA

Beispiel 1:

Annahme:

MP	KO	MO	ASSIDCODE	WA	Wert
201903	ISICQSE060216	89006VS	QOXDBM015510	ICAP	50
201903	ISICQSE060216	89006VS	PREMIQAHOLD	ICAP	50

Aufruf:

pruefWKN(<ISICQSE060216>, ASSIDCODE)

Ergebnis:

QOXDBM015510 ist ein gültiges Wertpapier in OBServ → True

PREMIQAHOLD ist kein gültiges Wertpapier in OBServ → False

3.10.12 checkBez

Diese Funktion prüft, ob eine bestimmte Beziehung zwischen zwei Identnummern vorliegt und liefert in diesem Fall TRUE bzw. FALSE zurück.

Funktionsaufruf:

checkBez(#Param1, #Param2, #Param3, #Param4, #Param5, #Param6, #Param7, #Param8)

Parameterbeschreibung:

1. Parameter #Param1
   ist ein IS-Konzept, OS-Konzept, Rechenregel oder Syntax
2. Parameter #Param2
   ist eine Ident- oder MO-Dimension (Passiv-Ident)
3. Parameter #Param3
   ist eine Ident- oder MO-Dimension (Aktiv-Ident)
4. Parameter #Param4
   ist ein boolscher Wert, der angibt, ob die Richtung ignoriert wird oder nicht
5. Parameter #Param5
   ist eine Objektart
6. Parameter #Param6 (optional)
   ist eine Funktionsspalte nach der gefiltert werden soll
7. Parameter #Param7 (optional)
   ist ein Funktionswert, nach dem gefiltert werden soll. Es können mehrere Werte unter Hochkomma angegeben werden
8. Parameter #Param8 (optional)
   gibt eine Referenzperiode (Param8) an. Damit wird die Beziehung mit den Stammdaten der jeweiligen Referenzperiode abgeglichen. Es können relative (z.B. MP=+1M) oder absolute Werte (z.B. MP=202012) angegeben werden.

Datentypkompatibilität:

Datentyp (Eingabe)	Ergebnis
Number	TRUE/FALSE
String	TRUE/FALSE
Boolean	TRUE/FALSE
Date	TRUE/FALSE

NaN	NaN
NA	ignoreNa = True → NA wird zu TRUE/FALSE ignoreNa = False → NA wird zu NA
#NR	TRUE/FALSE
Number as String	TRUE/FALSE
Boolean as String	TRUE/FALSE
Date as String	TRUE/FALSE
Leere Menge	ignoreNa = True → leere Menge wird zu TRUE/FALSE ignoreNa = False → leere Menge wird zu NA

Beispiel:

checkBez(IS01, INC, IN, False, swiftsz, Funktion, "90, 92", MP=+1M)

3.10.13 contains

Die Funktion contains prüft, ob eine Dimensionskombination in einem weiteren Konzept enthalten ist und liefert in diesem Fall True bzw. False zurück.

Funktionsaufruf:

contains(#Param1, #Param2, #Param3)

Beispiel:

contains(<IS01>, <IS02>, MO, LD)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder ein String.

Parameter #Param2 ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder ein String.

Parameter #Param3 ist eine Liste von Dimensionen(1 - n)

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Boolean
String	Boolean
Boolean	Boolean
Date	Boolean
NaN	NaN
NA	ignoreNa = True → NA wird zu Boolean ignoreNa = False → NA wird zu NA
#NR	Boolean
Number as String	Boolean
Boolean as String	Boolean
Date as String	NA
Leere Menge	ignoreNa = True → leere Menge wird zu Boolean ignoreNa = False → leere Menge wird zu NA

Beispiele:

Ausgangskonzept:

<IS01>

LD	WG	Wert
AT	EUR	1
AT	USD	2
DE	EUR	3
PL	USD	4

<IS02>

LD	WG	IN	Wert
AT	EUR	123	1
AT	USD	123	2
UK	EUR	123	3
PL	NOK	123	4

Beispiel 1:

Aufruf:

contains(<IS01>,<IS02>,LD,WG)

Ergebnis:

LD	WG	Wert
AT	EUR	True
AT	USD	True
DE	EUR	False
PL	USD	False

Beispiel 2:

Aufruf:

contains(<IS01>,<IS02>,LD)

Ergebnis:

LD	WG	Wert
AT	EUR	True
AT	USD	True
DE	EUR	False
PL	USD	True

Beispiel 3:

Aufruf:

contains(<IS01>,<IS02>,WG)

Ergebnis:

LD	WG	Wert
AT	EUR	True
AT	USD	True
DE	EUR	True
PL	USD	True

3.11 Stammdaten-Funktionen – Intervallarithmetik

3.11.1 crossJoin – Intervallarithmetik

Funktionsbeschreibung

Bei diesem Join werden die Dimensionen von Konzepten aufgefüllt.

Funktionsaufruf

crossJoin(#Konzept1, #Konzept2, "[#Dimliste]")

Beispiel:

crossJoin(<IS01>,<IS02>,"[MO,MP]")

Parameterbeschreibung

- #Konzept1, #Konzept2 sind Konzepte (Skalar/Vektor)
- #Dimliste sind die Dimensionen, anhand derer gejoint wird

Prüfung

- #Konzept1 und #Konzept2 müssen vorhandene IS oder OS-Konzepte sein
- Dimensionen in der #Dimliste müssen Dimensionen sowohl von #Konzept1 und #Konzept2 sein

Annahme:

<IS01> besteht aus MO, LD und hat oBdA nur eine Zeile

MO	LD	Wert
1200MO	AT	[500; 1500]

<IS02> besteht aus MO, WG und hat oBdA nur eine Zeile

MO	WG	Wert
1200MO	EUR	A

Der Funktonsaufruf wäre crossJoin(<IS01>, <IS02>, "[MO]").

Das Ergebnis soll folgendermaßen aussehen:

Im ersten Schritt werden die Konzepte zusammengehängt und die Spalten vergrößert

MO	LD	WG	Wert
1200MO	AT		[500; 1500]

1200MO		EUR	A

Im zweiten Schritt werden die Dimensionen erweitert.

MO	LD	WG	Wert
1200MO	AT	EUR	[500; 1500]
1200MO	AT	EUR	A

Keine Eindeutige Dimensionen:

Sollten die restlichen Dimensionen (ohne der #DimListe) nicht eindeutig sein, ist kein Fehler auszugeben. Es sind die Zeilen zu verdoppeln.

z.B:

<IS01>

MO	LD	FK	Wert
1200MO	AT	S0000	[500; 1500]
1200MO	DE	S0100	[2500; 3500]

<IS02>

MO	WG	FK	Wert
1200MO	EUR	S0200	A

crossJoin(<IS01>, <IS02>, "[MO]").

MO	LD	FK	WG	Wert
1200MO	AT	S0000	EUR	[500; 1500]
1200MO	DE	S0100	EUR	[2500; 3500]
1200MO	?	S0200	EUR	A

wegen der nicht-Eindeutigkeit von LD von <IS01> kann LD von <IS02> nicht aufgefüllt werden. Somit sollen die Zeilen verdoppelt und aufgefüllt werden.

MO	LD	FK	WG	Wert
1200MO	AT	S0000	EUR	[500; 1500]
1200MO	AT	S0100	EUR	[2500; 3500]
1200MO	AT	S0200	EUR	A
1200MO	DE	S0200	EUR	A

Sonderfall: nur eine Zeile vorhanden

<IS01>

MO	LD	WG	Wert
1200MO	AT	EUR	[500; 1500]

<IS02>

MO	WG	Wert
1200MO	EUR	A

crossJoin(<IS01>,<IS02>,"[MO,WG]")

MO	LD	WG	Wert
1200MO	AT	EUR	[500; 1500]

Es sollte hier als Standardverhalten der erste Wert genommen werden, also der Wert von<IS01> sollte hier im Ergebnis stehen.

Beispiele

Beispiel 1

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Wert
1200MO	AT	[500; 1500]
3400MO	DE	[5500; 6500]

5600MO	DE	[4500; 5500]
5600MO	AT	[1500; 2500]

<IS02>

MO	WG	Wert
1200MO	EUR	A
3400MO	EUR	B
5600MO	EUR	C
5600MO	USD	D

Aufruf

aggExc(<IS01>,<IS02>,"[MO]")

Ergebnis

MO	LD	WG	Wert
1200MO	AT	EUR	[500; 1500]
3400MO	DE	EUR	[5500; 6500]
5600MO	DE	EUR	[4500; 5500]
5600MO	AT	EUR	[1500; 2500]
5600MO	DE	USD	[4500; 5500]
5600MO	AT	USD	[1500; 2500]

Beispiel 2

<IS01>

MO	LD	FK	Wert
1200MO	AT	S0000	[500; 1500]
1200MO	DE	S0100	[2500; 3500]

<IS02>

MO	WG	FK	Wert
1200MO	EUR	S0200	A

Aufruf

aggExc(<IS01>,<IS02>,"[MO]")

Ergebnis

MO	LD	FK	WG	Wert
1200MO	AT	S0000	EUR	[500; 1500]
1200MO	AT	S0100	EUR	[2500; 3500]
1200MO	AT	S0200	EUR	A
1200MO	DE	S0200	EUR	A

Beispiel 3

<IS01>

MO	LD	WG	Wert
1200MO	AT	EUR	[500; 1500]

<IS02>

MO	WG	Wert
1200MO	EUR	A

Aufruf

crossJoin(<IS01>,<IS02>,"[MO,WG]")

Ergebnis

MO	LD	WG	Wert
1200MO	AT	EUR	[500; 1500]

3.12 String-Funktionen

Alle String-Funktionen werden in Prüfungen verwendet.

3.12.1 Umgang mit NA

Bei String-Funktionen (Kapitel String-Funktionen und asChar) wird bei der Verwendung von ignoreNa NA mit "" (Leer-String) ersetzt.

3.12.2 toUpper

Die Funktion toUpper gibt eine Zeichenfolge in Großbuchstaben zurück. Ist der mitgegebene Wert eine Zahl wird der Typ des Werts von BigDecimal in String gecastet.

Funktionsaufruf:

toUpper(Konzept)

Parameterbeschreibung:

1. Parameter Konzept

- ist ein IS-Konzept oder OS-Konzept

Beispiel 1:

Annahme:

KO	MO	MP	Wert
IS01	30937	31.01.2017	abc
IS01	30000	31.01.2017	aBc
IS01	40000	31.01.2017	AbC
IS01	50000	31.01.2017	Abc

Aufruf:

toUpper (<IS01>)

Ergebnis**:**

KO	MO	MP	Wert
IS01	30937	31.01.2017	ABC
IS01	30000	31.01.2017	ABC
IS01	40000	31.01.2017	ABC
IS01	50000	31.01.2017	ABC

Beispiel 2:

Annahme:

KO	MO	MP	Wert
IS01	30937	31.01.2017	23
IS01	30000	31.01.2017	12

Aufruf:

toUpper (<IS01>)

Ergebnis**:**

KO	MO	MP	Wert
IS01	30937	31.01.2017	"23"
IS01	30000	31.01.2017	"12"

3.12.3 toLower

Die Funktion toLower gibt eine Zeichenfolge in Kleinbuchstaben zurück. Ist der mitgegebene Wert eine Zahl wird der Typ des Werts von BigDecimal in String gecastet.

Funktionsaufruf:

toLower(Konzept)

Parameterbeschreibung:

1. Parameter Konzept

- ist ein IS-Konzept oder OS-Konzept

Beispiel 1:

Annahme:

KO	MO	MP	Wert
IS01	30937	31.01.2017	abc
IS01	30000	31.01.2017	aBc
IS01	40000	31.01.2017	AbC
IS01	50000	31.01.2017	Abc

Aufruf:

toLower(<IS01>)

Ergebnis**:**

KO	MO	MP	Wert
IS01	30937	31.01.2017	abc
IS01	30000	31.01.2017	abc
IS01	40000	31.01.2017	abc
IS01	50000	31.01.2017	abc

Beispiel 2:

Annahme:

KO	MO	MP	Wert
IS01	30937	31.01.2017	23
IS01	30000	31.01.2017	12

Aufruf:

toLower(<IS01>)

Ergebnis**:**

KO	MO	MP	Wert
IS01	30937	31.01.2017	"23"
IS01	30000	31.01.2017	"12"

3.12.4 inStr

Gibt die Position einer Zeichenfolge (case-sensitiv) innerhalb einer anderen Zeichenfolge als Zahl zurück (unter Berücksichtigung eines optionalen start-Parameters). Ist die gesuchte Zeichenfolge in der zu durchsuchenden Zeichenfolge nicht enthalten, wird -1 zurückgegeben.

Funktionsaufruf:

inStr(Konzept, suchStr, start)

Parameterbeschreibung:

1. Parameter Konzept ist ein IS-Konzept oder OS-Konzept
2. Parametr suchStr: die gesuchte Zeichenfolge (case sensitive)
3. Parameter start: ist ein optionaler Parameter, der die Startposition der Suche festlegt

Annahme:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	"Das ist ein schöner WertText"

Beispiel 1:

Aufruf:

inStr(<ISDEMO_EINWOHNER>, "Wert")

Ergebnis:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	21

Beispiel 2:

Aufruf:

inStr(<ISDEMO_EINWOHNER>, "wert")

Ergebnis**:**

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	-1

Beispiel 3:

Aufruf:

inStr(<ISDEMO_EINWOHNER>, "s")

Ergebnis**:**

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	3

Beispiel 4:

Aufruf:

inStr(<ISDEMO_EINWOHNER>, "s", 10)

Ergebnis**:**

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	13

3.12.5 replace

Mit der Funktion replace wird in einem Vektor jedes Vorkommen eines Wertes mit einem anderen Wert ersetzt. Dies gilt auch für Character- oder String-Werte.

Bei der Verwendung von "NA" (NA as String) ist folgendes zu berücksichtigen:

- Wenn bei #Param2 und #Param3 "NA" verwendet wird, wird zuerst das ignoreNa angewandt (hat keine Auswirkung) und anschließend das "NA" zu NA umgewandelt und anschließend das replace durchgeführt.
- Wenn bei #Param1 "NA" verwendet wird, bleibt dies "NA".

Funktionsaufruf:

replace(Param1, Param2, Param3)

Beispiel:

replace(<IS01>, 4, 6)

Parameterbeschreibung:

1. Parameter ist ein Vektor
2. Parameter ist der zu ersetzende Wert
3. Parameter ist der Ersatzwert

Datentypkompatibilität des Ersatzwertes:

Datentyp (Eingabe)	Ergebnis
Number	Number
String	String
Boolean	Boolean
Date	Date
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 ignoreNa = False → NA wird zu NA
#NR	#NR
Number as String	Number as String
Boolean as String	Boolean as String
Date as String	Date as String

Beispiele:

Ausgangsdaten

<IS01>

LD	Wert
AT	5
DE	5
NL	8
FR	NA

Beispiel 1:

Aufruf

replace(<IS01>,5,1)

Ergebnis

LD	Wert
AT	1
DE	1
NL	8
FR	NA

Beispiel 2:

Aufruf

replace(<IS01>,Na,0)

Ergebnis

LD	Wert
AT	5
DE	5
NL	8
FR	0

Weitere Beispiele:

Aufruf	Ergebnis
replace(5, 5, NA, "ignoreNa")	0
replace(5, 5, "NA", "ignoreNa")	NA
replace(5, 5, "NA")	NA
replace(NA, 0, 5, "ignoreNa")	5
replace(NA, 0, 5)	NA
replace("NA", "NA", 5)	"NA" (String)
replace(NA, NA, 5, "ignoreNa")	5
replace(NA, "NA", 5, "ignoreNa")	0
replace(NA, NA, 5)	5
replace(NA, "NA", 5)	5

3.12.6 strReplace

Ersetzt eine Zeichenfolge innerhalb einer anderen Zeichenfolge und gibt das Ergebnis als String zurück. Optional können die start-Position der Suche und die Anzahl der gewünschten Ersetzungen angegeben werden. Ist die gesuchte Zeichenfolge in der zu durchsuchenden Zeichenfolge nicht enthalten, wird die zu durchsuchende Zeichenfolge unverändert zurückgegeben.

Funktionsaufruf:

strReplace(Konzept, suchStr, ersatzStr, start, Anzahl)

Parameterbeschreibung:

1. Parameter Konzept
- ist ein IS-Konzept oder OS-Konzept
2. Parameter suchStr
- die zu ersetzende Zeichenfolge (case sensitive)
3. Parameter ersatzStr
- ersetzende Zeichenfolge
4. Parameter start
- ist ein optionaler Parameter, der die Startposition der Suche festlegt
5. Parameter Anzahl
- Ebenso optional; Präzisiert die Anzahl der zu ersetzenden Zeichenfolgen (default – alle)

Annahme:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	Das ist ein schöner WertText

Beispiel 1:

Aufruf:

strReplace(<ISDEMO_EINWOHNER>, "WertText", "Test")

Ergebnis:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	Das ist ein schöner Test

Beispiel 2:

Aufruf:

strReplace(<ISDEMO_EINWOHNER>, "s", "X")

Ergebnis:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	DaX iXt ein Xchöner WertText

Beispiel 3:

Aufruf:

strReplace(<ISDEMO_EINWOHNER>, "s", "X", 3, 1)

Ergebnis:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	Das iXt ein schöner WertText

3.12.7 subStr

Gibt einen Teil einer Zeichenfolge als String zurück.

Funktionsaufruf:

subStr(Konzept, start, stop)

Parameterbeschreibung:

1. Parameter Konzept
- ist ein IS-Konzept oder OS-Konzept
2. Parameter start
- Erstes Zeichen (beginnend mit 1)
3. Parameter stop
- optional: Letztes Zeichen (default – bis Ende des Strings)

Annahme:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	Das ist ein schöner WertText

Beispiel 1:

Aufruf:

1. subStr(<ISDEMO_EINWOHNER>, 8)
2. subStr(<ISDEMO_EINWOHNER>, 1, 3)
3. subStr(<ISDEMO_EINWOHNER>, 13, 17)

Ergebnis**:**

Bsp	KO	MO	MP	Wert
1	ISDEMO_EINWOHNER	2615	31.12.2015	ein schöner WertText
2	ISDEMO_EINWOHNER	2615	31.12.2015	Das
3	ISDEMO_EINWOHNER	2615	31.01.2017	schön

Beispiel 2:

Annahme:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2013	4799252

Aufruf:

subStr(<ISDEMO_EINWOHNER>, 1, 3)

Ergebnis:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2013	"479"

3.12.8 nChar

Gibt die Länge einer Zeichenfolge als Integer zurück.

Funktionsaufruf:

nChar(Konzept)

Parameterbeschreibung:

1. Parameter Konzept: ist ein IS-Konzept oder OS-Konzept

Annahme:

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	Das ist ein schöner WertText
ISDEMO_EINWOHNER	2615	31.12.2015	

Beispiel 1:

Aufruf:

nChar(<ISDEMO_EINWOHNER>)

Ergebnis**:**

KO	MO	MP	Wert
ISDEMO_EINWOHNER	2615	31.12.2015	28
ISDEMO_EINWOHNER	2615	31.12.2015	0

3.13 String-Funktionen - Intervallarithmetik

3.13.1 replace – Intervallarithmetik

Funktionsbeschreibung

Mit der Funktion replace wird in einem Vektor jedes Vorkommen eines Wertes mit einem anderen Wert ersetzt. Dies gilt auch für Character- oder String-Werte.

Bei der Verwendung von "NA" (NA as String) ist folgendes zu berücksichtigen:

- Wenn bei #Param2 und #Param3 "NA" verwendet wird, wird zuerst das ignoreNa angewandt (hat keine Auswirkung) und anschließend das "NA" zu NA umgewandelt und anschließend das replace durchgeführt.
- Wenn bei #Param1 "NA" verwendet wird, bleibt dies "NA".

Funktionsaufruf

replace(#Param1, #Param2, #Param3)

Parameterbeschreibung

- #Param1 ist ein Vektor
- #Param2 ist der zu ersetzende Wert
- #Param3 ist der Ersatzwert

Datentypkompatibilität des Ersatzwertes

Datentyp (Eingabe)	Ergebnis
Number	Number
String	String
Boolean	Boolean
Date	Date
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 ignoreNa = False → NA wird zu NA
#NR	#NR
Number as String	Number as String
Boolean as String	Boolean as String
Date as String	Date as String

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte	Intervall
1200MO	AT	1000	[500; 1500]
3400MO	DE	6000	[5500; 6500]
5600MO	DE	5000	[4500; 5500]
5600MO	AT	2000	[1500; 2500]

<IS02>, @decimals=-3

MO	LD	Werte	Intervall
1200MO	AT	1000	[500; 1500]
3400MO	DE	4000	[3500; 4500]
5600MO	DE	5500	[5000; 6000]
5600MO	AT	NA	NA

Beispiel 1

Aufruf

replace(<IS01>,<IS01>,10)

Ergebnis

MO	LD	Werte
1200MO	AT	[10; 10]
3400MO	DE	[10; 10]
5600MO	DE	[10; 10]
5600MO	AT	[10; 10]

Beispiel 2

Aufruf

replace(<IS01>,<IS02>,10)

Ergebnis

MO	LD	Werte
1200MO	AT	[10; 10]
3400MO	DE	[5500; 6500]
5600MO	DE	[4500; 5500]
5600MO	AT	[1500; 2500]

Beispiel 3

Aufruf

replace(<IS01>, 1000, 10)

Ergebnis

MO	LD	Werte
1200MO	AT	[500; 1500]
3400MO	DE	[5500; 6500]
5600MO	DE	[4500; 5500]
5600MO	AT	[1500; 2500]

3.13.2 subStr – Intervallarithmetik

Funktionsbeschreibung

Die Funktion subStr soll den String aus dem Wertefeld beginnend beim Zeichen in Position #Param1 bis Zeichen in Position #Param2 rückgeben. (Zählung der Positionen beginnt bei 1) Ist #Param2 nicht vorhanden, soll der String ab Position #Param1 zurückgegeben werden.

Funktionsaufruf

subStr(#String, #Param1, #Param2, #ParamX)

Beispiel:

subStr(<IS01>, 1,3)

Parameterbeschreibung

- Parameter String: Ist ein IS-Konzept, OS-Konzept, Rechenregel oder eine ANUBIS-Syntax die einen Vektor zurückgibt

- Parameter Param1: Position des Startzeichens des rückzugebenden Strings in #Vektor
- Parameter Param2: (optional) Position des Endzeichens des rückzugebenden Strings in #Vektor

Prüfung

- #String muss ein Vektor sein
- #Param1 muss größer 0 sein
- #Param2 ist optional und muss bei Vorhandensein größer 0 sein.
- #ParamX ist optional und kann "ignoreNA" sein.

Algorithmus

Die Funktion subStr soll den String aus dem Wertefeld beginnend beim Zeichen in Position #Param1 bis Zeichen in Position #Param2 rückgeben. (Zählung der Positionen beginnt bei 1) Ist #Param2 nicht vorhanden, soll der String ab Position #Param1 zurückgegeben werden.

Beispiele

Ausgangskonzept:

<IS01>

MO	Werte
1200MO	"Test"
1400MO	1000
3400MO	"6000"
5600MO	"NA"
1700MO	NA

Beispiel 1

Aufruf:

subStr(<IS01>,1,3)

Ergebnis:

MO	Werte
1200MO	"Tes"
1400MO	"100"
3400MO	"600"

5600MO	"NA"
1700MO	NA

Beispiel 2

Aufruf:

subStr(<IS01>,1,3,"ignoreNA")

Ergebnis:

MO	Werte
1200MO	"Tes"
1400MO	"100"
3400MO	"600"
5600MO	"NA"
1700MO	""

Beispiel 3

Aufruf:

subStr(<IS01>,3,1,"ignoreNA")

Ergebnis:

Fehler (Start>Ende)

Beispiel 4

Aufruf:

subStr(<IS01>,2)

Ergebnis:

MO	Werte
1200MO	"est"
1400MO	"000"
3400MO	"000"
5600MO	"A"

1700MO	NA

Beispiel 5

Aufruf:

subStr(<IS01>,2,1000)

Ergebnis:

MO	Werte
1200MO	"est"
1400MO	"000"
3400MO	"000"
5600MO	"A"
1700MO	NA

(Ende>Länge → Ende wird auf Länge des Strings gesetzt)

Beispiel 6

Aufruf:

subStr(<IS01>,0,3,"ignoreNA")

Ergebnis:

Fehler (Start<1)

Beispiel 7

Aufruf:

subStr(<IS01>)

Ergebnis:

Fehler (Zu wenige Parameter)

3.14 Datentypprüfungen

3.14.1 matchRegEx

Die Funktion matchRegEx prüft, ob Werte mitgelieferten Regular Expression Patterns entsprechen, also diese "matchen".

Funktionsaufruf:

Die Funktion kann mit 2 oder 3 Parametern aufgerufen werden.

Aufruf mit 2 Parametern: **matchRegEx(Param1, Param2)**

Aufruf mit 3 Parametern (vektorisierter Aufruf): **matchRegEx(Param1, Param2, Param3)**

Beispiel:

Aufruf mit 2 Parameter

LD=AT → Aufruf: matchRegEx(getDim(KonzeptXYZ,"PLZ"),"\d{4}")

Aufruf mit 3 Parameter

matchRegEx(getDim(KonzeptXYZ,"PLZ"),
mapping(getDim(KonzeptXYZ,"LD"),MP_LD_PLZ_REGEX),"True")

Parameterbeschreibung:

bei 2 Parametern:

1. Parameter: Ist ein OS/IS-Konzept bzw. eine Rechenregel oder ein Eingabewert, welche im Wertefeld den zu überprüfenden Wert enthält.
2. Parameter: Ist ein Stringwert welcher die Regular Expression beinhaltet z.B.: "[0-9]{4}[A-Z]{2}[0-9]{4}[A-Z]{2}".

bei 3 Parametern

1. Parameter: Ist ein OS/IS-Konzept bzw. eine Rechenregel, welche im Wertefeld den zu überprüfenden Wert enthält.
2. Parameter: Ist ein OS/IS-Konzept bzw. eine Rechenregel, welche im Wertefeld die anzuwendende Regular Expression Definition enthält. (Hier ist die gleiche Dimensionalität notwendig zur Überprüfung von Wertfeld aus Param1 und Wertfeld aus Param2)
3. Parameter: Ist ein Fixwert, entweder "True" oder "False" oder "NA" welcher das Defaultverhalten angibt, wenn kein passender Wert zur Dimensionskombination von Param1 in Param2 gefunden wird

Datentypkompatibilität:

Datentyp	Ergebnis
Number	True / False / NA

String	True / False / NA
Boolean	True / False / NA
Date	True / False / NA
NaN	NA
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu NA
#NR	NA
Number as String	True / False / NA
Boolean as String	True / False / NA
Date as String	True / False / NA
Leere Menge	ignoreNa = True → leere Menge wird zu False ignoreNa = False → leere Menge wird zu NA

Beispiele:

Ausgangskonzepte:

getDim(KonzeptXYZ,"PLZ"):

MP	EC	MO	LD	PLZ	Wert
20200131	GKE1	1200MO	AT	1230	1230
20200131	GKE1	1200MO	DE	88099	88099
20200131	GKE1	1200MO	NL	9104BR	9104BR
20200131	GKE1	3400MO	AT	A-1230	A-1230
20200131	GKE1	3400MO	DE	880991	880991
20200131	GKE1	3400MO	NL	91045R	91045R
20200131	GKE1	1700MO	BF	123xyz	123xyz
20200131	GKE1	1700MO	AT	#NR	NA

KonzeptREGEX (= mapping(getDim(KonzeptXYZ,"LD"),MP_LD_PLZ_REGEX))

MP	EC	MO	LD	PLZ	Wert
20200131	GKE1	1200MO	AT	1230	\d{4}
20200131	GKE1	1200MO	DE	88099	\d{5}
20200131	GKE1	1200MO	NL	9104BR	[0-9]{4}[A-Z]{2}
20200131	GKE1	3400MO	AT	A-1230	\d{4}
20200131	GKE1	3400MO	DE	880991	\d{5}
20200131	GKE1	3400MO	NL	91045R	[0-9]{4}[A-Z]{2}
20200131	GKE1	1700MO	BF	123xyz	NA
20200131	GKE1	1700MO	AT	#NR	\d{4}

Mapping: MP_LD_PLZ_REGEX

LD	Wert
AT	\d{4}
DE	\d{5}
NL	[0-9]{4}[A-Z]{2}
Defaultwert	NA

KonzeptXYZ_ohne_LAND:

MP	EC	MO	PLZ	Wert
20200131	GKE1	1200MO	1230	1230
20200131	GKE1	1200MO	88099	88099
20200131	GKE1	1200MO	9104BR	9104BR
20200131	GKE1	3400MO	A-1230	A-1230
20200131	GKE1	3400MO	880991	880991
20200131	GKE1	3400MO	91045R	91045R

20200131	GKE1	1700MO	123xyz	123xyz
20200131	GKE1	1700MO	#NR	NA

Beispiel 1 (mit 2 Parameter):

Aufruf:

matchRegEx(getDim(KonzeptXYZ,"PLZ"),"[0-9]{4}[A-Z]{2}")

Ergebnis:

MP	EC	MO	LD	PLZ	Wert
20200131	GKE1	1200MO	AT	1230	False
20200131	GKE1	1200MO	DE	88099	False
20200131	GKE1	1200MO	NL	9104BR	True
20200131	GKE1	3400MO	AT	A-1230	False
20200131	GKE1	3400MO	DE	880991	False
20200131	GKE1	3400MO	NL	91045R	False
20200131	GKE1	1700MO	BF	123xyz	False
20200131	GKE1	1700MO	AT	#NR	False

Beispiel 2 (mit 3 Parameter):

Aufruf:

matchRegEx(getDim(KonzeptXYZ,"PLZ"),
mapping(getDim(KonzeptXYZ,"LD"),MP_LD_PLZ_REGEX),"True")

Ergebnis:

MP	EC	MO	LD	PLZ	Wert	*Erklärung*
20200131	GKE1	1200MO	AT	1230	True	
20200131	GKE1	1200MO	DE	88099	True	
20200131	GKE1	1200MO	NL	9104BR	True	
20200131	GKE1	3400MO	AT	A-1230	False	
20200131	GKE1	3400MO	DE	880991	False	

20200131	GKE1	3400MO	NL	91045R	False	
20200131	GKE1	1700MO	AT	#NR	False	
20200131	GKE1	1700MO	BF	123xyz	True	*wegen Param 3: Defaultverhalten, wenn kein passender Wert zur Dimensionskombination von Param1 in Param2 gefunden wird*

Beispiel 3 (mit 3 Parameter) (ignoreNa=True):

Aufruf:

```
matchRegEx(getDim(KonzeptXYZ,"PLZ"),
mapping(getDim(KonzeptXYZ,"LD"),MP_LD_PLZ_REGEX),"True")
```

Ergebnis:

MP	EC	MO	LD	PLZ	Wert	*Erklärung*
20200131	GKE1	1200MO	AT	1230	True	
20200131	GKE1	1200MO	DE	88099	True	
20200131	GKE1	1200MO	NL	9104BR	True	
20200131	GKE1	3400MO	AT	A-1230	False	
20200131	GKE1	3400MO	DE	880991	False	
20200131	GKE1	3400MO	NL	91045R	False	
20200131	GKE1	1700MO	AT	#NR	False	
20200131	GKE1	1700MO	BF	123xyz	True	*wegen Param 3: Defaultverhalten wenn kein passender Wert zur Dimensionskombination von Param1 in Param2 gefunden wird*

3.14.2 isNa

Die Funktion isNa prüft, ob die mitgeschickte Variable einen NA-Wert enthält. Ist dieser vorhanden wird ein True geliefert. Ansonsten wird ein False geliefert.

Funktionsaufruf:

isNa(#Param1)

Beispiel:

isNa(x)

Parameterbeschreibung:

#Param1 ist ein numerischer/boolscher Skalar/Vektor

Datentypkompatibilität:

Datentyp (Eingabe)	Ergebnis
Number	False
String	False
Boolean	False
Date	False
NaN	False
NA	ignoreNa = True → NA wird zu 0 → False ignoreNa = False → NA wird zu NA → True
#NR	False
Number as String	False
Boolean as String	False
Date as String	False
Leere Menge	ignoreNa = True → Leere Menge wird zu NA. NA wird zu 0 → False ignoreNa = False → Leere Menge wird zu NA → True

Beispiele:

Ausgangskonzept

<IS01>

LD	Wert
AT	123
DE	NA
FR	456

Beispiel 1:

Aufruf

isNa(<IS01>)

Ergebnis

LD	Wert
AT	False
DE	True
FR	False

Beispiel 2:

Aufruf

ignoreNA = True

isNa(<IS01>)

Ergebnis:

LD	Wert
AT	False
DE	False
FR	False

3.14.3 isChar

Die Funktion isChar gibt True zurück, wenn es sich bei der geprüften Variable um einen Character(String) handelt. Bei allen anderen Variablentypen ist das Ergebnis ein False.

Funktionsaufruf:

isChar(#Param1)

Beispiel:

isChar(<IS01>)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder ein String.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	False
String	True
Boolean	False
Date	False
NaN	False
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu False
#NR	True
Number as String	True
Boolean as String	True
Date as String	True
Leere Menge	ignoreNa = True → leere Menge wird zu False ignoreNa = False → leere Menge wird zu False

Beispiele:

Ausgangskonzept:

<IS01>

MP	Wert
20200131	5.000.000,00
20200131	Musterbank 1
20200131	NA
20200131	NaN

Beispiel 1:

Aufruf:

isChar(<IS01>)

Ergebnis:

MP	Wert
20200131	False
20200131	True
20200131	False
20200131	False

Beispiel 2:

Aufruf:

isChar(<IS01>, „ignoreNa“)

Ergebnis:

MP	Wert
20200131	False
20200131	True
20200131	False
20200131	False

3.14.4 isLogical

Die Funktion isLogical gibt True zurück, wenn es sich bei der geprüften Variable um einen booleschen Wert (True, False) handelt. Bei allen anderen Variablentypen ist das Ergebnis ein False.

Funktionsaufruf:

isLogical(#Param1)

Beispiel:

if(!isLogical(<IS01>);1;0)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder ein String.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	False
String	False
Boolean	True
Date	False
NaN	False
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu False
#NR	False
Number as String	False
Boolean as String	False
Date as String	False
Leere Menge	ignoreNa = True → leere Menge wird zu False ignoreNa = False → leere Menge wird zu False

Beispiele:

Ausgangskonzept:

<IS01>

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	NA
3400MO	AT	123
3400MO	DE	#NR

Beispiel 1:

Aufruf:

isLogical(<IS01>)

Ergebnis:

MO	LD	Wert
1200MO	AT	True
1200MO	DE	True
1200MO	NL	False
3400MO	AT	False
3400MO	DE	False

Beispiel 2:

Aufruf:

if(!isLogical(<IS01>);1;0)

Ergebnis:

MO	LD	Wert
1200MO	AT	0
1200MO	DE	0
1200MO	NL	1
3400MO	AT	1
3400MO	DE	1

3.14.5 isNumeric

Die Funktion isNumeric gibt True zurück, wenn es sich bei der geprüften Variable um einen numerischen Wert handelt. Bei allen anderen Variablentypen ist das Ergebnis ein False.

Funktionsaufruf:

isNumeric (#Param1)

Beispiel:

isNumeric(<IS01>)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, ein Eingabewert oder ein String.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	True
String	False
Boolean	False
Date	False
NaN	False
NA	ignoreNa = True → NA wird zu True ignoreNa = False → NA wird zu False
#NR	False
Number as String	False
Boolean as String	False
Date as String	False
Leere Menge	ignoreNa = True → leere Menge wird zu True ignoreNa = False → leere Menge wird zu False

Beispiele:

Ausgangskonzept:

<IS01>

MO	LD	Wert
1200MO	AT	True
1200MO	DE	-12000
1200MO	NL	NA
3400MO	AT	123
3400MO	DE	#NR

Beispiel 1:

Aufruf:

isNumeric (<IS01>)

Ergebnis:

MO	LD	Wert
1200MO	AT	False
1200MO	DE	True
1200MO	NL	False
3400MO	AT	True
3400MO	DE	False

Beispiel 2:

Aufruf:

if(!isNumeric(<IS01>);1;0)

Ergebnis:

MO	LD	Wert
1200MO	AT	1
1200MO	DE	0
1200MO	NL	1
3400MO	AT	0
3400MO	DE	1

3.15 Datentypprüfungen - Intervallarithmetik

3.15.1 isNa – Intervallarithmetik

Funktionsbeschreibung

Die Funktion isNa prüft, ob die mitgeschickte Variable einen NA-Wert enthält. Ist dieser vorhanden wird ein True geliefert. Ansonsten wird ein False geliefert.

Funktionsaufruf

isNa(#Param1)

Beispiel:

isNa(x)

Parameterbeschreibung

- #Param1 ist ein numerischer/boolscher Skalar/Vektor

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	False
String	False
Boolean	False
Date	False
NaN	False
NA	ignoreNa = True → NA wird zu 0 → False ignoreNa = False → NA wird zu NA → True
#NR	False
Number as String	False
Boolean as String	False
Date as String	False
Leere Menge	ignoreNa = True → Leere Menge wird zu NA. NA wird zu 0 → False ignoreNa = False → Leere Menge wird zu NA → True

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	6000
5600MO	DE	NA
5600MO	AT	2000

Beispiel 1

Aufruf

isNa(<IS01>)

Ergebnis

MO	LD	Werte
1200MO	AT	False
3400MO	DE	False
5600MO	DE	True
5600MO	AT	False

3.16 Konvertierungs-Funktionen

3.16.1 asChar

Die Funktion asChar wandelt alle Werte eines Eingangsvektors in einen Ausgangsvektor bestehend aus Strings um. Ein skalarer Wert wird in einen String umgewandelt und retourniert.

Funktionsaufruf:

asChar(#Param1)

Beispiel:

asChar(<IS01>)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder ein String.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	String
String	String
Boolean	String
Date	String
NaN	NaN
NA	ignoreNa = True → NA wird zu leere Menge ignoreNa = False → NA wird zu NA
#NR	#NR
Number as String	String
Boolean as String	String
Date as String	String
Leere Menge	ignoreNa = True → leere Menge wird zu leere Menge ignoreNa = False → leere Menge wird zu leere Menge

Beispiele:

Ausgangskonzept:

<IS01>

MO	Wert
1200MO	-12
3400MO	True
5600MO	False
7800MO	0
9900MO	NA
2200MO	"2200MO"

Beispiel 1:

Aufruf:

asChar(<IS01>)

Ergebnis:

MO	Wert
1200MO	"-12"
3400MO	"True"
5600MO	"False"
7800MO	"0"
9900MO	NA
2200MO	"2200MO"

Beispiel 2:

Aufruf:

asChar(<IS01>, "ignoreNa")

Ergebnis:

MO	Wert
1200MO	"-12"

3400MO	"True"
5600MO	"False"
7800MO	"0"
9900MO	
2200MO	"2200MO"

3.16.2 asLogical

Der Hauptzweck der Funktion asLogical ist einen numerischen Vektor in einen logischen Vektor umzuwandeln (x=0 → False, x!=0 → True). Die Funktion kann weiters mit den Datentypen Boolean, Boolean as String, Date (wird wie eine Zahl interpretiert) und Number as String umgehen.

Funktionsaufruf:

asLogical(#Param1)

Beispiel:

asLogical(<IS01>)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder ein String.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Boolean
String	NA
Boolean	Boolean
Date	Boolean
NaN	NaN
NA	ignoreNa = True → NA wird zu False ignoreNa = False → NA wird zu NA
#NR	NA
Number as String	Boolean
Boolean as String	Boolean
Date as String	NA
Leere Menge	ignoreNa = True → leere Menge wird zu leere Menge ignoreNa = False → leere Menge wird zu leere Menge

Beispiele:

Ausgangskonzept:

<IS01>

MO	Wert
1200MO	-12
3400MO	True
5600MO	False
7800MO	0
9900MO	NA
1100MO	0,001
2200MO	2200MO

Beispiel 1:

Aufruf:

asLogical(<IS01>)

Ergebnis:

MO	Wert
1200MO	True
3400MO	True
5600MO	False
7800MO	False
9900MO	NA
1100MO	True
2200MO	NA

Beispiel 2:

Aufruf:

asLogical(<IS01>, "ignoreNa")

Ergebnis:

MO	Wert
1200MO	True
3400MO	True
5600MO	False
7800MO	False
9900MO	False
1100MO	True
2200MO	NA

3.16.3 asNumeric

Die Funktion asNumeric wandelt alle Werte eines Eingangsvektors in einen Ausgangsvektor bestehend aus numerischen Werten um. Ein skalarer Wert wird in einen numerischen Wert umgewandelt und retourniert.

Funktionsaufruf:

asNumeric(#Param1)

Beispiel:

asNumeric(<IS01>)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder String.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number
String	NA
Boolean	Number
Date	Number
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 ignoreNa = False → NA wird zu NA
#NR	NA
Number as String	Number
Boolean as String	Number
Date as String	Number
Leere Menge	ignoreNa = True → leere Menge wird zu leere Menge ignoreNa = False → leere Menge wird zu leere Menge

Beispiele:

Ausgangskonzept:

<IS01>

MO	Wert
1200MO	-12
3400MO	"1234"
3500MO	"#NR"
5600MO	False
7800MO	True
9900MO	NA
2200MO	"2200MO"

Beispiel 1:

Aufruf:

asNumeric(<IS01>)

Ergebnis:

MO	Wert
1200MO	-12
3400MO	1234
3500MO	NA
5600MO	0
7800MO	1
9900MO	NA
2200MO	NA

Beispiel 2:

Aufruf:

asNumeric(<IS01>, "ignoreNa")

Ergebnis:

MO	Wert
1200MO	-12
3400MO	1234
3500MO	NA
5600MO	0
7800MO	1
9900MO	0
2200MO	NA

3.16.4 unaryNot

Die unaryNot-Funktion negiert logische boolean Werte. Numerische Werte werden zuerst in boolsche Werte gecastet (0 wird zu False und numerischen Werte ungleich 0 zu True) und anschließend logisch negiert.

Funktionsaufruf:

unaryNot(#Param1)

Beispiel:

unaryNot(<IS01>)

!<IS01>

Parameterbeschreibung:

Parameter #Param1 ist ein gültiges OS-Konzept, IS-Konzept, eine Rechenregel oder ein Eingabewert

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Boolean
String	techn. Fehler
Boolean	Boolean
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu True ignoreNa = False → NA wird zu NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu True ignoreNa = False → leere Menge wird zu NA

Beispiele:

Ausgangskonzept:

<IS01>

MO	LD	Wert
1200MO	AT	True
1200MO	DE	False
1200MO	NL	NA
3400MO	AT	-123
3400MO	ES	0

Beispiel 1:

Aufruf:

!<IS01>

Ergebnis:

MO	LD	Wert
1200MO	AT	False
1200MO	DE	True
1200MO	NL	NA
3400MO	AT	False
3400MO	ES	True

Beispiel 2:

Aufruf:

unaryNot(<IS01>, "ignoreNa")

Ergebnis:

MO	LD	Wert
1200MO	AT	False
1200MO	DE	True

1200MO	NL	True
3400MO	AT	False
3400MO	ES	True

3.16.5 mapping

Es werden mit Hilfe einer Mapping-Tabelle die Vektorwerte eines Konzepts auf die Zielwerte gemappt. Bei nur einer Zielwert-Spalte erfolgt der Funktionsaufruf mit zwei Parametern. Bei mehreren Zielwert-Spalten gibt der dritte Parameter an, auf welche Spalte gemappt werden soll (Matrix-Mapping).

Als zu mappende Werte (Ausgangswerte) können Intervalle angegeben werden. Diese sind in Klammern zu schreiben. "[" bedeutet inklusiv und "(" bedeutet exklusiv.

Wenn NA als Ausgangswert vorkommt, ignoreNa=False ist und es keinen zu mappenden Zielwert gibt, ist das Ergebnis ein NA (nicht der Default-Wert). Wenn ignoreNa=True ist, wird NA zu 0 und entsprechend gemappt.

Als Default-Ziel-Wert ist der Datentyp Number, String, NA as String, NaN as String, NA (Exceptional Value) und NaN (Exceptional Value) möglich. Als Zielwert in der Mapping-Tabelle ist nur der Datentyp Number und String möglich.

Funktionsaufruf:

Einfaches Mapping: Aufruf mit 2 Parametern: mapping(#Param1, #Param2)

Matrix-Mapping: Aufruf mit 3 Parametern: mapping(#Param1, #Param2, #Param3)

Beispiel:

mapping(getDim(<IS01>, LD), MPLAENDER1)

mapping(getDim(<IS01>, LD), MPLAENDER2, SPALTEDT)

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel oder Eingabewert

Parameter #Param2 ist die Mapping-ID

Parameter #Param3 ist die Spalten-ID (optional, nur bei Matrix Mapping)

Datentypkompatibilität:

Datentyp	Ergebnis
Number	hängt vom Zielwert ab
String	hängt vom Zielwert ab
Boolean	hängt vom Zielwert ab
Date	hängt vom Zielwert ab
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 → 0 wird gemappt. Ergebnis hängt vom Mapping-Zielwert ab. ignoreNa = False → NA wird zu NA

#NR	hängt vom Zielwert ab
Number as String	hängt vom Zielwert ab
Boolean as String	hängt vom Zielwert ab
Date as String	hängt vom Zielwert ab
Leere Menge	ignoreNa = True → leere Menge wird zu leere Menge ignoreNa = False → leere Menge wird zu leere Menge

Beispiele:

Ausgangskonzept:

<IS01>

KO	LD	Wert
EINWOHNER	AT	8
EINWOHNER	DE	79
EINWOHNER	FR	64
EINWOHNER	ES	46
EINWOHNER	JP	NA

Mapping: MP01

Default-Mapping: NA (Exceptional Value)

Ausgangswert	A	B
AT	Österreich	AUT
DE	Deutschland	DEU
NA	Y	Y
0	Z	Z

Mapping: MP02

Default-Mapping: Wert = X

Ausgangswert	Zielwert
[0,50)	A
[50,70]	B

Mapping: MP03

Default-Mapping: Wert = X

Ausgangswert	Zielwert
0	5
String	neuer String
True	10
NaN	30
NA	40
#NR	50
10	False
20	NaN
30	NA
40	#NR

Beispiel 1:

Aufruf:

mapping(getDim(<IS01>, LD), MP01, A)

Ergebnis:

KO	LD	Wert
EINWOHNER	AT	Österreich
EINWOHNER	DE	Deutschland
EINWOHNER	FR	NA

EINWOHNER	ES	NA
EINWOHNER	JP	NA

Beispiel 2:

Aufruf:

mapping(getDim(<IS01>, LD), MP01, A, "ignoreNa")

Ergebnis:

KO	LD	Wert
EINWOHNER	AT	Österreich
EINWOHNER	DE	Deutschland
EINWOHNER	FR	NA
EINWOHNER	ES	NA
EINWOHNER	JP	NA

Beispiel 3:

Aufruf:

mapping(<IS01>, MP02)

Ergebnis:

KO	LD	Wert
EINWOHNER	AT	A
EINWOHNER	DE	X
EINWOHNER	FR	B
EINWOHNER	ES	A
EINWOHNER	JP	NA

Beispiel 4:

Aufruf:

mapping(<IS01>, MP02, „ignoreNa“)

Ergebnis:

KO	LD	Wert
EINWOHNER	AT	A
EINWOHNER	DE	X
EINWOHNER	FR	B
EINWOHNER	ES	A
EINWOHNER	JP	A

Weitere Beispiele:

Aufruf	Typ Ausgangswert	Ergebnis	Typ Ergebnis	Kommentar
mapping(0, MP03)	Number	5	Number	
mapping("String", MP03)	String	neuer String	String	
mapping(True, MP03)	Boolean	X	String	
mapping("True", MP03)	String	10	Number	
mapping(NaN, MP03)	Exceptional Value NaN	NaN	Exceptional Value NaN	
mapping("NaN", MP03)	String	NaN	Exceptional Value NaN	"NaN" wird zu NaN gecastet.
mapping(NA, MP03)	Exceptional Value NA	NA	Exceptional Value NA	
mapping(NA, MP03, "ignoreNa")	NA	5	Number	NA wird zu 0 gecastet und 0 wird zu 5 gemappt.
mapping("NA", MP03)	String	NA	Exceptional Value NA	"NA" wird zu NA gecastet. Es erfolgt kein Mapping.
mapping("NA", MP03, "ignoreNa")	String	5	Number	"NA" wird zu NA gecastet und NA zu 0 und 0 wird zu 5 gemappt.

mapping("#NR", MP03)	String	50	Number	
mapping(10, MP03)	Number	False	String	
mapping(20, MP03)	Number	NaN	String	
mapping(30, MP03)	Number	NA	String	
mapping(30, MP03, "ignoreNa")	Number	NA	String	
mapping(40, MP03)	Number	#NR	String	

3.16.6 abs

Funktionsbeschreibung

Berechnet den Absolutbetrag der übergebenen Werte.

Funktionsaufruf

abs(#Param1)

Beispiel:

abs(-3)

Parameterbeschreibung

#Param1 ist ein IS-Konzept, OS-Konzept, eine Rechenregel oder ein Eingabewert

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 0 ignoreNa = False → NA bleibt NA

#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler
Leere Menge	ignoreNa = True → leere Menge wird zu 0 ignoreNa = False → leere Menge bleibt NA

Beispiele

Ausgangskonzept:

<IS01>

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	-600
5600MO	DE	NA
5600MO	AT	-200

Beispiel 1

Aufruf

abs(<IS01>)

Ergebnis

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	600
5600MO	DE	NA
5600MO	AT	200

Beispiel 2

Aufruf

abs(<IS01>, “ignoreNa”)

Ergebnis

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	600
5600MO	DE	0
5600MO	AT	200

3.17 Konvertierungs-Funktionen - Intervallarithmetik

3.17.1 asChar – Intervallarithmetik

Funktionsbeschreibung

Die Funktion asChar wandelt alle Werte eines Eingangsvektors in einen Ausgangsvektor bestehend aus Strings um. Ein skalarer Wert wird in einen String umgewandelt und retourniert.

Funktionsaufruf

asChar(#Param1)

Beispiel:

asChar(<IS01>)

Parameterbeschreibung

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder ein String.

Datentypkompatibilität

Datentyp	Ergebnis
Number	String
String	String
Boolean	String
Date	String
NaN	NaN
NA	ignoreNa = True → NA wird zu Leere Menge ignoreNa = False → NA wird zu NA
#NR	#NR
Number as String	String
Boolean as String	String
Date as String	String
Leere Menge	ignoreNa = True → leere Menge wird zu leere Menge

	ignoreNa = False → leere Menge wird zu leere Menge

Beispiele

Ausgangskonzept

<IS01>

MO	Wert
1200MO	-12
3400MO	True
5600MO	False
7800MO	0
9900MO	NA
2200MO	"2200MO"

Beispiel 1

Aufruf

asChar(<IS01>)

Ergebnis

MO	Wert
1200MO	"-12"
3400MO	"True"
5600MO	"False"
7800MO	"0"
9900MO	NA
2200MO	"2200MO"

Beispiel 2

Aufruf

asChar(<IS01>, "ignoreNa")

Ergebnis

MO	Wert
1200MO	"-12"
3400MO	"True"
5600MO	"False"
7800MO	"0"
9900MO	""
2200MO	"2200MO"

3.17.2 asNumeric – Intervallarithmetik

Funktionsbeschreibung

Die Funktion asNumeric wandelt alle Werte eines Eingangsvektors in einen Ausgangsvektor bestehend aus numerischen Werten um. Ein skalarer Wert wird in einen numerischen Wert umgewandelt und retourniert.

Funktionsaufruf

asNumeric(#Param1)

Beispiel:

asNumeric(<IS01>)

Parameterbeschreibung

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder String.

Datentypkompatibilität

Datentyp	Ergebnis
Number	Number
String	NA
Boolean	Number
Date	Number
NaN	NaN
NA	ignoreNa = True → NA wird zu 0

	ignoreNa = False → NA wird zu NA
#NR	NA
Number as String	Number
Boolean as String	Number
Date as String	Number
Leere Menge	ignoreNa = True → leere Menge wird zu Leere Menge ignoreNa = False → leere Menge wird zu Leere Menge

Beispiele:

Ausgangskonzept:

<IS01>

MO	LD	Werte
1200MO	AT	"1000"
3400MO	DE	"6000"
5600MO	DE	NA
5600MO	AT	[2000; 2000]

Beispiel 1

Aufruf:

asNumeric(<IS01>)

Ergebnis:

MO	LD	Werte
1200MO	AT	[1000; 1000]
3400MO	DE	[6000; 6000]

5600MO	DE	NA
5600MO	AT	[2000; 2000]

3.17.3 abs – Intervallarithmetik

Funktionsbeschreibung

Berechnet den Absolutbetrag der übergebenen Werte. Bei Intervallen wird der Absolutbetrag der Intervallmitte genommen werden und das Toleranzintervall mit der gleichen Breite um diese gesetzt.

Funktionsaufruf

abs(#Param1)

Beispiel:

abs(<IS01>)

Parameterbeschreibung

#Param1 ist ein IS-Konzept, OS-Konzept, eine Rechenregel oder ein Eingabewert

Datentypkompatibilität

Datentyp (Eingabe)	Ergebnis
Number	Number
String	techn. Fehler
Boolean	techn. Fehler
Date	techn. Fehler
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA bleibt NA
#NR	techn. Fehler
Number as String	techn. Fehler
Boolean as String	techn. Fehler
Date as String	techn. Fehler

Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge bleibt NA

Algorithmus

Berechnet den Absolutbetrag der übergebenen Werte. Bei Intervallen wird die Intervallmitte sowie die Breite des Intervalls berechnet. Anschließend wird der Absolutbetrag der Intervallmitte genommen und ein gleich breites Intervall um diese, neue Intervallmitte gesetzt.

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	Werte
1200MO	AT	1000
3400MO	DE	-6000
5600MO	DE	5000
5600MO	AT	-200

Beispiel 1

Aufruf

abs(<IS01>)

Ergebnis

MO	LD	Werte
1200MO	AT	[500; 1500]
3400MO	DE	[5500; 6500]
5600MO	DE	[4500; 5500]
5600MO	AT	[-300; 700]

Beispiel 2

Aufruf

abs(<IS01>,"ignoreInterval")

Ergebnis

MO	LD	Werte
1200MO	AT	[1000; 1000]
3400MO	DE	[6000; 6000]
5600MO	DE	[5000; 5000]
5600MO	AT	[200; 200]

3.18 Mengenfunktionen

Werden nicht in Prüfungen verwendet.

3.18.1 intersection

Wird nicht in Prüfungen verwendet.

Die Funktion intersection berechnet die Schnittmenge zweier Mengen, um jene Dimensionskombinationen zu identifizieren, die in beiden Mengen vorkommen.

Funktionsaufruf:

intersection(#Konzept1, #Konzept2)

Parameterbeschreibung:

1. Parameter #Konzept1
   ist ein IS-Konzept, ein OS-Konzept oder eine Rechenregel
2. Parameter #Konzept2
   ist ein IS-Konzept, ein OS-Konzept oder eine Rechenregel

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Dimensionskombination, Wert "1"
String	Dimensionskombination, Wert "1"
Boolean	Dimensionskombination, Wert "1"
Date	Dimensionskombination, Wert "1"
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA wird zu NA
#NR	#NR (Dimensionskombinationen mit NULL-Wert fließen in den Vergleich nicht ein, #NR ist aber valide Dimensionsausprägung)
Number as String	Dimensionskombination, Wert "1"
Boolean as String	Dimensionskombination, Wert "1"

Date as String	Dimensionskombination, Wert "1"
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge wird zu NA

3.18.2 union

Wird nicht in Prüfungen verwendet.

Die Funktion union berechnet die Vereinigung zweier Mengen, um jene Dimensionskombinationen zu identifizieren, die in der ersten, der zweiten oder in beiden Mengen vorkommen.

Funktionsaufruf:

union(#Konzept1, #Konzept2)

Parameterbeschreibung:

1. Parameter #Konzept1
   ist ein IS-Konzept, ein OS-Konzept oder eine Rechenregel
2. Parameter #Konzept2
   ist ein IS-Konzept, ein OS-Konzept oder eine Rechenregel

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Dimensionskombination, Wert "1"
String	Dimensionskombination, Wert "1"
Boolean	Dimensionskombination, Wert "1"
Date	Dimensionskombination, Wert "1"
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA wird zu NA
#NR	#NR (Dimensionskombinationen mit NULL-Wert fließen in den Vergleich nicht ein, #NR ist aber valide Dimensionsausprägung)
Number as String	Dimensionskombination, Wert "1"
Boolean as String	Dimensionskombination, Wert "1"
Date as String	Dimensionskombination, Wert "1"
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge wird zu NA

3.18.3 *difference*

Wird nicht in Prüfungen verwendet.

Die Funktion difference berechnet die Differenzmenge zweier Vektoren, um jene Dimensionskombinationen zu identifizieren, die nur im ersten Parameter-Vektor vorkommen.

Funktionsaufruf:

difference(#Param1, #Param2)

Beispiel:

difference(<IS01>, <IS02>) → gibt die Differenzmenge der Konzepte <IS01> und <IS02> zurück

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel

Parameter #Param2 ist ein IS-Konzept, OS-Konzept, Rechenregel

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Dimensionskombination, Wert "1"
String	Dimensionskombination, Wert "1"
Boolean	Dimensionskombination, Wert "1"
Date	Dimensionskombination, Wert "1"
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA wird zu NA
#NR	#NR (Dimensionskombinationen mit NULL-Wert fließen in den Vergleich nicht ein, #NR ist aber valide Dimensionsausprägung)
Number as String	Dimensionskombination, Wert "1"
Boolean as String	Dimensionskombination, Wert "1"
Date as String	Dimensionskombination, Wert "1"
Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge wird zu NA

3.18.4 symmDifference

Wird nicht in Prüfungen verwendet.

Die Funktion symmDifference berechnet die symmetrische Differenzmenge zweier Vektoren, um jene Dimensionskombinationen zu identifizieren, die jeweils nur in einem der beiden Parameter-Vektoren vorkommen.

Funktionsaufruf:

symmDifference(#Param1, #Param2)

Beispiel:

symmDifference(<IS01>,<IS02>) → gibt die symmetrische Differenzmenge der Konzepte <IS01> und <IS02> zurück

Parameterbeschreibung:

Parameter #Param1 ist ein IS-Konzept, OS-Konzept, Rechenregel

Parameter #Param2 ist ein IS-Konzept, OS-Konzept, Rechenregel

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Dimensionskombination, Wert "1"
String	Dimensionskombination, Wert "1"
Boolean	Dimensionskombination, Wert "1"
Date	Dimensionskombination, Wert "1"
NaN	NaN
NA	ignoreNa = True → NA wird zu 1 ignoreNa = False → NA wird zu NA
#NR	#NR (Dimensionskombinationen mit NULL-Wert fließen in den Vergleich nicht ein, #NR ist aber valide Dimensionsausprägung)
Number as String	Dimensionskombination, Wert "1"
Boolean as String	Dimensionskombination, Wert "1"
Date as String	Dimensionskombination, Wert "1"

Leere Menge	ignoreNa = True → leere Menge wird zu 1 ignoreNa = False → leere Menge wird zu NA

3.19 Weitere Funktionen

3.19.1 filterResult

Die Funktion filterResult soll den Wertevektor auf die in der Liste angegebenen Dimensionen einschränken. In der Prüfregelsyntax gibt es drei verschiedene Dimensionseinschränkungen.

1. Einzel - LD=AT
2. Gruppen
   a. SCS-Sektoren - LD_G=BSEU
   b. eckige Klammern - LD_G=[AT, DE, PL]
3. Dimensionen - LD_D=LDC
4. oben abgeführte Beispiel auch als NOT (! – LD!=AT)

Funktionsaufruf:

filterResult(#Vektor, #Filterliste)

Parameterbeschreibung:

- Parameter #Vektor ist ein IS-Konzept, OS-Konzept, Rechenregel oder eine Prüfregel-Syntax die einen Vektor zurückgibt
- Parameter #Filterliste wird als Liste (durch Kommata getrennt, in Hochkomma) an Einschränkungen eingegeben (z.B. LD=AT, WG_G=BS01, LD_D=LDC). Dabei kann auf die Werte-Spalte gefiltert werden. Für Werte sind die folgenden Vergleichsoperatoren zulässig: >, <, <=, >=, ==, !=. Zusätzlich ist es möglich auf den NA-Wert zu filtern. Der Parameter kann 1 bis n mal verwendet werden.

Beispiele:

Ausgangskonzept:

Konzept	LD	LDC	Wert
IS01123	AT	AT	0
IS01123	DE	DE	1
IS01123	PL	US	1
IS01123	US	CU	2

IS01123	AT	PL	3
IS01123	CU	US	5
IS01123	KW	KW	8

Beispiel 1

Aufruf:

filterResult (<IS01123>, “LD=AT”)

Ergebnis:

Konzept	LD	LDC	Wert
IS01123	AT	AT	0
IS01123	AT	PL	3

Beispiel 2:

Aufruf:

filterResult (<IS01123>, “LD=AT”, “LDC=AT”)

Ergebnis:

Konzept	LD	LDC	Wert
IS01123	AT	AT	0

Beispiel 3:

Aufruf:

filterResult (<IS01123>, “LD_G=BSEU”)

Ergebnis:

Konzept	LD	LDC	Wert
IS01123	AT	AT	0
IS01123	DE	DE	1
IS01123	PL	US	1
IS01123	AT	PL	3

Beispiel 4:

Aufruf:

filterResult (<IS01123>, “LD_G=BSEU”, “LDC=US”)

Ergebnis:

Konzept	LD	LDC	Wert
IS01123	PL	US	1

Beispiel 5:

Aufruf:

filterResult (<IS01123>, “LD_G=BSEU”, “LDC_G=BSEU”)

Ergebnis:

Konzept	LD	LDC	Wert
IS01123	AT	AT	0
IS01123	DE	DE	1
IS01123	AT	PL	3

Beispiel 6:

Aufruf:

filterResult (<IS01123>, “LD_D=LDC”)

Ergebnis:

Konzept	LD	LDC	Wert
IS01123	AT	AT	0
IS01123	DE	DE	1
IS01123	KW	KW	8

Beispiel 7:

Aufruf:

filterResult (<IS01123>, “LD_G=[AT,DE]”)

Ergebnis:

Konzept	LD	LDC	Wert
IS01123	AT	AT	0

IS01123	DE	DE	1
IS01123	AT	PL	3

Beispiel 8:

Filterung des IS-Konzepts und danach Filterung des Resultats (macht in diesem Bsp. wenig Sinn, da man gleich das IS-Konzept filtern könnte, soll aber das Prinzip verdeutlichen, dass der benötigte Vektor auf verschiedene Arten und Weisen generiert werden können soll, z.B. Zusammenrechnen von 2 gefilterten Konzepten, crossJoin, Rechenregel usw.)

Aufruf:

filterResult (<IS01123 LD!=AT>, “LD_D=LDC”)

Ergebnis:

Konzept	LD	LDC	Wert
IS01123	DE	DE	1
IS01123	KW	KW	8

Weitere Funktionsaufrufe:

Aufruf mit einem CrossJoin (CrossJoin hat als Rückgabe immer einen Vektor):

filterResult(crossJoin(<ISSHSG_0102_WPI>, <ISSHSG_0301_WPI>,"[WK]"), “LD_D=LDC”)

Aufruf mit einer Rechenregel (die genannte Rechenregel hat als Rückgabe einen Vektor):

filterResult(RRSHSGJN_0102WPI_0301WPI, “LD_D=LDC”)

3.19.2 *length*

Die Funktion length berechnet die Anzahl der Werte.

Die Funktion kann mit 1 bis N Parametern ausgeführt werden.

Bei der Berechnung mit einem Parameter der ein Vektor ist, werden alle Werte des Vektors gezählt.

Bei der Berechnung mit 2 bis N Parametern mit mindestens einen Vektor wird zeilenweise gezählt.

Mit ignoreNa können NAs von der Zählung ausgenommen werden.

Funktionsaufruf:

length(#ParamN)

Beispiel:

length(<IS01>)

Parameterbeschreibung:

Parameter #ParamN ist ein IS-Konzept, OS-Konzept, Rechenregel, Eingabewert oder String.

Datentypkompatibilität:

Datentyp	Ergebnis
Number	Number
String	Number
Boolean	Number
Date	Number
NaN	Number
NA	ignoreNa = True → NA wird nicht berücksichtigt. ignoreNa = False → NA wird berücksichtigt.
#NR	Number
Number as String	Number
Boolean as String	Number
Date as String	Number
Leere Menge mit 2 bis N Parameter	ignoreNa = True → leere Menge wird zu leere Menge

(mindestens 1 Vektor)	ignoreNa = False → leere Menge wird zu leere Menge
Leere Menge mit 1 Parameter (Vektor)	ignoreNa = True → leere Menge wird zu 0 ignoreNa = False → leere Menge wird zu 0

Beispiele:

Ausgangskonzept:

<IS01>

MO	Wert
1200MO	-12
3400MO	NA
3500MO	"#NR"
5600MO	False

Beispiel 1:

Aufruf:

length(<IS01>)

Ergebnis:

4

Beispiel 2:

Aufruf:

length(<IS01>, "ignoreNa")

Ergebnis:

3

Beispiel 3:

Aufruf:

length(<IS01>, <IS01>, 3, "ignoreNa")

Ergebnis:

MO	Wert
1200MO	3

3400MO	1
3500MO	3
5600MO	3

3.20 Weitere Funktionen – Intervallarithmetik

3.20.1 filterResult – Intervallarithmetik

Funktionsbeschreibung

Die Funktion filterResult soll den Wertevektor auf die in der Liste angegebenen Dimensionen einschränken. Wenn in der Werte-Spalte Strings verwendet werden, dann werden diese, sobald Werte gefiltert werden, ignoriert.

Funktionsaufruf

filterResult(#Vektor, #Filterliste)

Beispiel:

filterResult (<IS01>, "WG=EUR", "LD=AT", "[Wert]>0")

Parameterbeschreibung

- Parameter #Vektor: Ist ein IS-Konzept, OS-Konzept, Rechenregel oder eine ANUBIS-Syntax die einen Vektor zurückgibt (z.B. ein crossJoin)
- Parameter #Filterliste: Wird als Liste (Beistrich-getrennt, in Hochkomma) an Einschränkungen eingegeben. Dabei kann auf die Werte-Spalte gefiltert werden. Für Werte sind die folgenden Vergleichsoperatoren zulässig: >, <, <=, >=, ==, !=. Zusätzlich ist es möglich auf den NA-Wert zu filtern.
  Der Parameter kann 1 bis n mal verwendet werden.

Prüfung

- #Vektor müssen vorhandene IS- oder OS-Konzepte, falls es IS- oder OS-Konzepte sind
- Die Einschränkungen in der #Filterliste müssen Dimensionen von #Vektor sein, falls #Vektor ein IS oder OS-Konzept ist.

Algorithmus

Die Funktion filterResult soll den Wertevektor auf die in der Liste angegebenen Dimensionen einschränken.

Beispiele

Ausgangskonzept:

<IS01>, @decimals=-3

MO	LD	WG	Werte	Intervall
1200MO	AT	EUR	1000	[500; 1500]
3400MO	DE	EUR	6000	[5500; 6500]
5600MO	DE	EUR	5000	[4500; 5500]

5600MO	AT	EUR	2000	[1500; 2500]
1200MO	AT	USD	5000	[4500; 5500]
3400MO	DE	USD	3000	[2500; 3500]
5600MO	DE	USD	10000	[9500; 10500]
5600MO	AT	USD	1000	[500; 1500]
1700MO	US	USD	7000	[6500; 7500]

Beispiel 1

Aufruf

filterResult (<IS01>, "LD=AT")

Ergebnis

MO	LD	WG	Werte
1200MO	AT	EUR	[500; 1500]
5600MO	AT	EUR	[1500; 2500]
1200MO	AT	USD	[4500; 5500]
5600MO	AT	USD	[500; 1500]

Beispiel 2

Aufruf

filterResult (<IS01>, "LD=AT", "WG=EUR")

Ergebnis

MO	LD	WG	Werte
1200MO	AT	EUR	[500; 1500]
5600MO	AT	EUR	[1500; 2500]

Beispiel 3

Aufruf

filterResult (<IS01>, "LD_G=BSBKAUSL", "WG=EUR")

(BSBKAUSL = EU ohne AT)

Ergebnis

MO	LD	WG	Werte
3400MO	DE	EUR	[5500; 6500]
5600MO	DE	EUR	[4500; 5500]

Beispiel 4

Aufruf

filterResult (<IS01>, "LD_G=[AT,DE]", "WG=EUR")

(BSBKAUSL = EU ohne AT)

Ergebnis

MO	LD	WG	Intervall
1200MO	AT	EUR	[500; 1500]
3400MO	DE	EUR	[5500; 6500]
5600MO	DE	EUR	[4500; 5500]
5600MO	AT	EUR	[1500; 2500]

Beispiel 5

Aufruf

filterResult (<IS01>, "[Wert]==1000")

(Anm. 1000 ist ein Skalar und wird als [1000; 1000] interpretiert)

Ergebnis

MO	LD	WG	Intervall
1200MO	AT	EUR	[500; 1500]
5600MO	AT	USD	[500; 1500]

Beispiel 6

Aufruf

filterResult (<IS01>, "[Wert]<1000")

(Anm. 1000 ist ein Skalar und wird als [1000; 1000] interpretiert)

Ergebnis

MO	LD	WG	Intervall
1200MO	AT	EUR	[500; 1500]
5600MO	AT	USD	[500; 1500]

3.21 Reservierte Schlüsselwörter

Die unterhalb aufgelisteten Bezeichnungen sind reservierte Schlüsselwörter der Prüfregelsyntax, die nicht zweckentfremdet werden dürfen. Diese reservierten Schlüsselwörter sind Case Insensitiv.

Name	Beschreibung
TRUE	eine zutreffende Aussage in der Aussagenlogik
FALSE	eine nicht zutreffende Aussage in der Aussagenlogik
NA	kennzeichnet ein Fehlen von Informationen bzw. Daten an einer Stelle
NAN	entspricht einem undefinierten oder nicht darstellbaren Wert

3.22 OeNB-interne Funktionen

Die folgenden Funktionen werden ausschließlich OeNB-intern verwendet. Falls eine der folgenden Funktionen in einer Prüfung vorkommt, dann bitten wir um eine Kontaktaufnahme.

Funktion	Kurzbeschreibung
bewertenWhg()	Die Werte eines (Währung-) Konzepts (Skalar/Vektor) werden mit einem Kurswert multipliziert.
bewertenWp()	Das Ergebnis des (Wertpapier-) Konzepts (Vektor/Skalar) wird mit dem durch die Kursart und Priorisierung (Zabil/Monstat) erzielten Kurs multipliziert.
blzToMo()	Ändert die Dimension MO von einer BLZ zu einer UUID.
calcRatio()	Die Funktion retourniert Zähler und Nenner in getrennten Datensätzen für eine gewichtete Division.
cos()	Berechnet den Cosinus von x (Möglichkeit "NA" zu ignorieren, Mit cos(x, ¨ignoreNa¨)).
entwertenWhg()	Die Werte eines (Währung-) Konzepts (Skalar/Vektor) werden mit einem Kurswert dividiert.
entwertenWp()	Das Ergebnis des (Wertpapier-) Konzepts (Vektor/Skalar) wird mit dem durch die Kursart und Priorisierung (Zabil/Monstat) erzielten Kurs dividiert.
errDivNeg()	Es wird die Division x/y ausgeführt, sollten dabei x und y negativ sein, wird als Ergebnis der Wert von z ausgegeben.
exp()	Berechnet e^x (Möglichkeit "NA" zu ignorieren; Mit exp(x, ¨ignoreNa¨)).
getAll()	Die Funktion getAll() liefert die Gesamtmenge aller zum Zeitpunkt der Abfrage gültigen Werte der IN oder MO Dimension.
getKonzessionen()	Die Funktion getKonzessionen() liefert die zu einer bestimmten Meldeperiode gültigen Konzessionen eines Idents und strukturiert sie mit Hilfe der im Funktionsaufruf angegebenen Dimensionen.
gnorm()	Berechnung der Inversen der Verteilungsfunktion einer normalverteilten Zufallsvariable mit Erwartungswert m und Standardabweichung s.
isLeMe()	Die Funktion isLeMe() soll rückliefern, ob zu einem Datensatz (z.B. aus einer Vorperiode) in der aktuellen Periode eine Leermeldung vorliegt.

lg()	Berechnet den Logarithmus von x zur Basis y. (Möglichkeit "NA" zu ignorieren; Mit lg(x,y, ¨ignoreNa¨)).
ln()	In Anlehnung an die Funktion exp() wird der natürliche Logarithmus von x berechnet (Möglichkeit "NA" zu ignorieren, Mit ln(x, ¨ignoreNa¨)).
moToBlz()	Diese Funktion wandelt die UUID bzw. IdentNr. der Meldeobjekt (MO) Dimension eines Konzepts in die BLZ um.
pnorm()	Berechnung des Wertes x der Normalverteilung einer Zufallszahl X (=P(X<x)) mit Erwartungswert m und Standardabweichung s.
quantile()	Berechnet das probs-ste Quantil. (Möglichkeit "NA, NULL, NaN" zu ignorieren, quantile(x, probs, ¨lastNa¨,¨ignoreNa¨ , ¨ignoreNull¨ , ¨ignoreNaN", "lastNa","firstNa"); Wobei mit "lastNa" die NAs am Schluss des Vektors gereiht werden (Null und NaN werden hier wie NA behandelt und daher auch in NA umgewandelt). Mit "firstNa" werden die NAs an den Anfang des Vektors gereiht (Null und NaN werden hier auch wie NA behandelt und daher auch in NA umgewandelt)).
rank()	Bewertet den Vektor x mittels unten angeführter Eigenschaften. Aufruf rank(x, ¨lastNa¨ , tiesMethod,¨increasing¨). lastNa...Reiht die verbleibenden NAs am Schluss des Vektors an. NaNs werden hier wie NA behandelt und daher auch in NA umgewandelt. firstNa...Reiht die verbleibenden NAs am Beginn des Vektors an. NaNs werden hier wie NA behandelt und daher auch in NA umgewandelt. tiesMethod...¨max¨, ¨min¨, kommt zum Tragen, wenn 2 oder mehrere Positionen im Vektor den gleichen Rang haben, default = ¨max¨ max ..... rank([5, 10, 10, 20, 30],¨max¨) Ergebnis: [1,3,3,4,5] min.....rank[10, 10, 20, 30],¨min¨) Ergebnis [1,1,3,4] ¨increasing¨.....Vektor wird ansteigend sortiert, Gegenteil ¨decreasing¨ default = ¨increasing¨;
reklassifikation()	Eine RR wird über eine implizite Dimension distinktiv sortiert (keine doppelten Zeilen). Die Zeilen des Ergebnisvektor werden dann mit ihrem Gegenpart aus der in der RR angegebenen Vorperiode an Hand der expliziten Dimensionen verglichen. Bei einem Unterschied wird eine Kopie der Zeile der Vorperiode eingefügt und in der angegebenen Zielwert-Dimension (zB WA=NO) (erfordert Bindung RR an TR!) die Reklassifizierung vermerkt.
sign()	Beschreibt das Vorzeichen der Zahl. (Möglichkeit "NA" zu ignorieren, Mit sign(x, ¨ignoreNa¨)).

sin()	Berechnet den Sinus von x (Möglichkeit "NA" zu ignorieren, Mit sin(x, ¨ignoreNa¨)).
std()	Berechnet die Standardabweichung des Vektors (Möglichkeit "NA, 0" zu ignorieren, Mit std(x, ¨ignoreNa¨, "ignoreZero")).
sqrt()	Berechnet die Wurzel von den Werten in x. (Möglichkeit "NA" zu ignorieren; Mit sqrt(x, ¨ignoreNa¨)).
tan()	Berechnet den Tangens von x (Möglichkeit "NA" zu ignorieren, Mit tan(x, ¨ignoreNa¨)).
transformieren()	Diese Funktion transformiert Daten (Parameter 1) mit der angegebenen Transformation (Parameter 2).
whgUmrechnen()	Diese Funktion soll ein Konzept mit Währungskursen (z.B. EUR bzw. Quellwährung) mit einem Währungskurs umrechnen.
winsor()	Ist eine Zahl des Vektors x größer/kleiner als der Schwellenwert, so wird dieses Element durch den Schwellenwert og/ug ersetzt. NA werden hier auch ersetzt.
wpKursbewertung()	Diese Funktion soll ein Konzept mit Wertpapierdaten (z.B. Stücken/Nominale) mit einem Wertpapierkurs be-/entwerten. "bewerten" bedeutet, dass mit dem jeweiligen Wertpapierkurs multipliziert und "entwerten" bedeutet, dass mit dem jeweiligen Wertpapierkurs dividiert wird.

4 Ansprechpartner

Bei ***fachlichen Fragen***, die die Prüfregel betreffen, wenden Sie sich bitte an die Erhebungsverantwortlichen.

Sie finden die Informationen unter

https://www.oenb.at/meldewesen.html

in der Sektion „Ansprechperson“.

Bei ***technischen Fragen***, die die Prüfregelsyntax betreffen, wenden Sie sich bitte an

Klemens Neubauer-Stiedl, BSc

Abteilung STADA

E-Mail: productownerHST@oenb.at