EIOPA Validations Syntax

The document contains non-binding information, and is subject to substantial further changes

LAST UPDATE: 28/05/2025

email: xbrl@eiopa.europa.eu ; Website: https://eiopa.europa.eu

INDEX

I Modification history

Date	Main change description
30/09/2015	First version of the document
15/07/2016	'rNNN' & 'cNNN' syntax added to the document. Replacing syntax 'for every' with 'not(isfallback)'
1/06/2017	'Reported' added to syntax
1/06/2017	'Allowed combinations of values'
1/06/2017	'Unit of a monetary concept for (…) does not match value of (…)'
1/11/2018	BV4 example for 'Like'/'not like' was replaced with BV6 validation (due to the fact that BV4 validation was removed from the 2.3.0 Hotfix scope)
3/06/2019	Updates to examples due to improvements in business and technical validations
15/07/2020	Updates to examples due to improvements in business and technical validations
15/07/2021	Updates to examples to make it more general
31/01/2023	Updates explaining new outputs and improved syntax (V2). Reorganisation of the file due to introduction of the new syntax.
28/05/2025	New example of conditional validation added.

II Introduction

Aim of this document is to describe syntax, wording and patterns used in definition of business and technical rules for EIOPA XBRL taxonomies. The examples provided are based on Solvency II validations.
Validations are presented using simplified expression syntax, aimed for business users, as well as more technical XBRL oriented one. Below document follows the prior structure.

III Syntax use cases

III.1 Generic mathematical and logical operators

Below table describes basic operators used in business rules

Operator	**Meaning**
+	Addition
-	Subtraction
*	Multiplication
/	Division
=	Equation
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
!=	Not equal (other than) (previously it was referenced as <>)
Sum()	Calculates summation of components inside the parenthesis
Max()	Finds maximum value from the components inside the parenthesis
Min()	Finds minimum value from the components inside the parenthesis
Abs()	Returns absolute value from the components inside the parenthesis
And	Both components must be true
Or	At least one component must be true
Exp()	Calculates the exponential function. It requires the expression, numerator and denominator inside parentheses separated by a comma
Count()	Counts the occurrence of a given fact

NOTE:

Although possible to implement, no validation currently uses absolute value operators. Furthermore, the use of the divide operator is avoided due to the risk of a divide-by-zero error. Instead, it was decided to reverse the equations to represent the described relationships by multiplication.

Some of the operators can be implemented with "i" prefix, indicating that the relation should be calculated using the interval arithmetic tolerance mechanism[1].

III.2 Syntax specific for EIOPA validations

Some syntax used in validations is specific for EIOPA project. Below particular case with explanation and examples are provided.

III.2.1 Data type constrains

Data type constrain is used to identify applicable patterns for a given reportable fact. In majority of the cases it refers to one of the ISO codification standards, like ISO 4217 for currencies or ISO 8601 for dates.

[1] The interval arithmetic is described in XBRL Taxonomy documentation, section VII.3.6.7 Evaluation of validation rules and interval arithmetic.

Examples:

Validation	Explanation
matches({t: S.23.04.01.01, c: C0110}, "yyyy-mm-dd pattern for date as per ISO 8601")	Value in column C0110 must be in line with ISO 8601 format (yyyy-mm-dd)
matches({t: S.23.04.01.04, c: C0470}, "one of options as per ISO 4217")	Value in column C0470 must be in line with ISO 4217 format (3 letter code for currencies)
matches({t: S.11.01.01.01, c: C0230}, "one of options as per ISO 3166-1, 'XA', 'EU' or 'AA'")	Value in column C0040 must be in line with ISO 3166-1 alpha-2 code format (2 letter code for country name), but additionally possible values are also "XA", "EU" and "AA" which do not belong to the standard
matches({t: S.01.02.01.01, r: R0050, c: C0010}, "one of options as per ISO 3166-1")	Value provided in row R0050 column C0010 must be in line with ISO 3166-1 alpha-2 code format (2 letter code for country name)
matches({t: S.01.02.01.01, r: R0070, c: C0010}, "one of options as per ISO 639-1")	Value in row R0070, column C0010 must be in line with ISO 639-1 alpha-2 code format (2 letter code for language),

NOTE:

This type of validation is represented in a taxonomy as a reference to list of domain members defined in the dictionary or by data type as an XML attribute, hence technically no XBRL formulas are generated for data type constrains.

III.2.2 Existence checks (previously 'Empty')

This type of validation checks whether particular reportable element was or was not reported. The operator used in these rules is "isNull" to indicate that the filed cannot be reported, and the prefix 'not' for checking the reverse. Another way to enforce reporting of a certain datapoint is use of count() function. In such case it is assumed that filer will have to provide a certain number of facts, corresponding to the number expressed in the equation.

Examples:

Validation	Explanation
not(isNull({t: E.04.01.16.01, r: ER0010}))	Cell er0010 reported in E.04.01.16.01, must not be empty
if not(isNull({t: S.08.01.01.01, c: C0240, z: Z0001})) then not(isNull({t: S.08.01.01.02, c: C0380, z: Z0001})) else true()	If column C0240 is reported, then column C0380 should also be reported
not(isNull({t: S.06.02.01.02, c: C0292, z: Z0001})); Where: matches({t: SE.06.02.18.02, z: Z0001, c: C0290, seq: False, id: v2, f: solvency, fv: solvency2}, "^..4.$")	Column c0292 has to be reported for CIC codes "##4#" (Investment funds Collective Investment Undertakings)

(not(isNull({t: S.12.01.02.01, r: R0030, c: C0190})) and not(isNull({t: S.12.01.02.01, r: R0100, c: C0190}))) or (isNull({t: S.12.01.02.01, r: R0030, c: C0190}) and isNull({t: S.12.01.02.01, r: R0100, c: C0190}))	Rows R0030 and R0100 for column C0190 in table S.12.01.02.01 should be either simultaneously reported or left empty
if ({t: S.26.05.04.05, r: R0010, c: C0010, z: Z0001} = [s2c_AP:x33] and not(isNull({t: S.05.01.01.01, r: R0200, c: C0100, z: Z0001}))) then not(isNull({t: S.26.05.04.01, r: R0160, c: C0060, z: Z0001})) and not(isNull({t: S.26.05.04.01, r: R0160, c: C0070, z: Z0001})) and not(isNull({t: S.26.05.04.01, r: R0160, c: C0090, z: Z0001})) else true()	If 'Simplifications used' is reported row R0010 column C0010 and R0200, C0100 for S.05.01.01.01 table is reported, then S.26.05.04.01 columns C0060, C0070 and C0090 for rows R0160 should be reported
if (not(isNull({t: S.06.03.01.01, c: C0060, z: Z0001}))) then not(isNull({t: S.06.02.01.02, c: C0290, z: Z0001})) else true()	If column c0600 is not empty, then column c0290 must also be reported
count({t: S.01.01.11.01, r: R0010; R0253; R0254; R0490; R0950; R0960; R0980, c: C0010, dv: emptySequence(), seq: True, id: v0, f: solvency, fv: solvency2}) = 7	There should be 7 facts reported for table S.01.01.11.01 rows R0010, R0253, R0254, R0490, R0950, R0960, R0980

III.2.3 Dictionary element reference

Since some of the reported facts are components of the dictionary (e.g. *s2c_SE:x10* which is an domain member from the SE domain and its label is *Undertakings pursuing both life and non-life insurance activity*), they are also used in a number of business rules. In the expressions, these cases are identified by putting relevant dictionary component within the square brackets. Dictionary references can be used as a part of the validation check (i.e., by requiring or prohibiting certain element to be reported), or as a part of additional constrain casted on specific variable (e.g., filter).

Examples:

Validation	**Explanation**
if ({t: S.01.02.04.01, r: R0190, c: C0010} = [s2c_AP:x10]) then {t:	If value reported in cell r0190,c0010 in table S.01.02 is *s2c_AP:x10* (*No use of transitional measure on the risk-free interest rate*), then Impact of transitional on

S.22.01.04.01, r: R0060, c: C0050} = 0 else true()	interest rate for Tier 1 reported (cell r0370,c0010) in table S.22.01.04.01 must equal 0
if ({t: S.14.01.01.01, c: C0030} = [s2c_LB:x10] or {t: S.14.01.01.01, c: C0030} = [s2c_LB:x11]) then isNull({t: S.14.01.01.05, c: C0060}) and isNull({t: S.14.01.01.05, c: C0061}) and isNull({t: S.14.01.01.05, c: C0062}) and isNull({t: S.14.01.01.05, c: C0063}) else true()	If a value reported on column C0030 (Total amount of Written premiums: of which written directly by the insurance undertaking) is *s2c_LB:x10* (Annuities stemming from non-life insurance contracts and relating to health insurance obligations) or s2c_LB:x11 (Annuities stemming from non-life insurance contracts and relating to insurance obligations other than health insurance obligations), then S.14.01.01.01 C0030 (Line of Business), S.14.01.01.05 C0061 (of which written directly by the insurance undertaking), C0062 (of which written via credit institutions), C0063 (of which written via other insurance distributors) should be empty
{ m: [s2md_met:ei1904]} != [s2c_CU:x7]	Value s2c_CU:x7 (Temporary identifier for currency) should not be reported for metric ei1904 (Metric: Swap delivered currency (for buyer))
dim({d: [s2c_dim:LR]},[s2c_dim:LR]) != [s2c_GA:x112]	The item "Temporary identifier for country 1" must not be reported for dimension LR regardless of the table
{t: S.17.01.01.01, r: R0160, c: C0090} = {t: S.19.01.01.04, r: R0260, c: C0360, filter: [s2c_dim:BL] = [s2c_LB:x34] and [s2c_dim:OC] = [s2c_CU:x0]}	Value reported in row R0160 (Gross - Total) for column C0090 (General liability insurance) reported in table S.17.01.01.01, should be equal to value reported in S.19.01.01.04 in row R0260 (Total) column C0360 (Year end (discounted data) 'General liability insurance [direct business and accepted proportional reinsurance]' for line of business
{t: S.06.02.01.01, c: C0060, z: Z0001, filter: not(isNull([s2c_dim:NF]))} = [s2c_PU:x96] or {t: S.06.02.01.01, c: C0060, z: Z0001, filter: not(isNull([s2c_dim:NF]))} = [s2c_PU:x57]	Column C0060 should be reported with value PU:x96 or PU:x57, in addition the Fund Number code must be provided. The validation assumes "Fund/Matching portfolio Number" value to be provided in S.06.02.01.01 to execute

III.2.4 'Matches'/'not matches' (previously 'Like / not like')

This operator provides mechanism to distinguish pattern or a given sign from the reported element. It is used primarily to filter out particular rows from open tables. In case of some validations using patterns they follow regular expression syntax and include '^' identifying beginning of a text and '$' identifying the end of a text.

Examples:

Validation	**Explanation**
if matches(dim({d: [s2c_dim:CA]},[s2c_dim:CA]), "^LEI/[A-Z0-9]{18}[0-9]{2}$") and not(matches(dim({d: [s2c_dim:CA]},[s2c_dim:CA]), "^LEI/[A-Z0-9]{18}(01\|00)$")) then leiChecksum(substring(dim({d: [s2c_dim:CA]},[s2c_dim:CA]),5))	If value reported for a Code Broker (s2c_dim:CA) starts with LEI followed by "/" followed later by ISIN code pattern, and it is not a test code with 01 or 00 final designs, then it should have correct checksum otherwise the Code Broker should be a Specific Code starting with "SC/"

else matches(dim({d: [s2c_dim:CA]},[s2c_dim:CA]), "^SC/.*") or isNull(dim({d: [s2c_dim:CA]},[s2c_dim:CA])) or matches(dim({d: [s2c_dim:CA]},[s2c_dim:CA]),[…]	
matches({ m: [s2md_met:si1554]}, "^....$")	Metric si1554 (Metric: String|TS/CIC code) should follow 4 sign pattern
if matches({ m: [s2md_met:si1559]}, "^LEI/[A-Z0-9]{20}$") and not(matches({ m: [s2md_met:si1559]}, "^LEI/[A-Z0-9]{18}(01\|00)$")) then leiChecksum(substring({ m: [s2md_met:si1559]}, 5)) else matches({ m: [s2md_met:si1559]}, "^None$")	Value reported for metric si1559 must be in line with LEI which is 20-character alphanumeric code, preceded by "LEI/". In addition, the code should have correct checksum. The only other accepted value is "None".

III.2.5 'Allowed combinations of values'

A specific example of the use of the matches function is the check of possible combinations of integers.

Validation	Explanation
matches({ m: [s2md_met:si2468], seq: False, id: v0}, "^((1$)\|(9$)){1}$")	The only values that can be reported for si2468 are "1" or "9"
matches({ m: [s2md_met:si2527]}, "^(((1$\|1,){0,1}(2$|2,){0,1}(3$\|3,){0,1}(4$|4,){0,1}$")	The only values that can be reported for si2527 are "1" or "2" or "3" or "4" or "1,2" or "1,3" or "1,4" or "2,3" or "2,4" or "3,4" or "1,2,3" or "1,2,4" or "1,3,4" or "2,3,4" or "1,2,3,4"

III.2.6 Conditional validations

These validations are represented by *If x then y* notation. Often, logical test (x) and the result if true (y) are complex expressions and are using other operators described in this document. Validation check is executed whether the if statement is met, in such case two possible outcomes are expected. Note however, that in case the if statement has been failed, validation will proceed with the last (i.e., else) part, which is usually set to true. As the result, report can be submitted without the fear of blocking validation being triggered for unforeseen scenario.

Examples:

Validation	Explanation
if ({t: S.26.01.04.03, r: R0020, c: C0010, z: Z0001} = [s2c_AP:x34] and isNull({t: S.02.01.01.01, r: R0850, c: C0010})) then {t: S.26.01.04.02, r: R0110, c: C0060, z: Z0001} = imax(0, ({t: S.26.01.04.01, r: R0110, c: C0020, z: Z0001} - {t: S.26.01.04.01, r: R0110, c:	If value reported in table S.26.01 cell r0020,c0010 is *s2c_AP:x34 (Simplifications not used)* and cell reported in S.02.01 r0850,c0010 is empty then, for table S.26.01, value in cell r0110,c0060 must be equal

C0030, z: Z0001}) - ({t: S.26.01.04.01, r: R0110, c: C0040, z: Z0001} - {t: S.26.01.04.01, r: R0110, c: C0050, z: Z0001})) else true()	to the maximum value of either 0 or result of subtracting r0110,c0030; r0110,c0040 and r0110,c0050 from r0110,c0020
if (matches(dim({d: [s2c_dim:IW]},[s2c_dim:IW]) , "^CAU/.*")) then (matches(dim({d: [s2c_dim:IW]},[s2c_dim:IW]), "^(CAU/MAL$)\|(CAU/INDEX/.+)\|(^CAU/INST/.+)\|(CAU/ISIN/[A-Z0-9]{11}\d\+[A-Z]{3}$)")) else true()	If value provided for typed dimension IW follow pattern starting with "CAU/" followed by none or infinite number of any characters, then it should start with "CAU/INST/" followed by any number of characters or "CAU/MAL" or "CAU/INDEX/" followed by at least one character or "CAU/INST/" followed by at least one character, or "CAU/ISIN/" followed by 20 alphanumerical ISIN code, with the last character being a digit, followed by three capital letters. In case value reported does not start with "CAU/" pattern, the validation will execute the else statement and pass as TRUE
if ({t: S.01.02.01.01, r: R0090, c: C0010} < createDate(2025, 12, 31)) then isNull({t: C.06.02.01.02, c: C0235}) else true()	If date provided in row R0090 is earlier than 2025/12/31 then column C0235 must not be reported

III.2.7 Scope (previously 'NNN' & 'cNNN')

The scope column is used to indicate to which rows and columns the validation is applicable. The field provides information for each variable referred in the validation expression.

Examples:

Validation	**Explanation**
scope({t: SR.22.03.01.01, r:R0070;R0080;R0090;R0100;R0110, f: solvency, fv: solvency2}) {t: SR.22.03.01.01, c: C0010, z: Z0001} >= 0	Value in cell c0010 rows R0070, R0080, R0090, R0100 and R0110 in table SR.22.03.01.01, should be positive or zero
scope({t: S.17.03.01.01, c:C0100, f: solvency, fv: solvency2},{t: S.17.03.01.02, c:C0100, f: solvency, fv: solvency2},{t: S.17.01.01.01, c:C0100, f: solvency, fv: solvency2}) {t: S.17.03.01.01, r: R0010} + {t: S.17.03.01.01, r: R0020} + {t: S.17.03.01.01, r: R0030} + sum({t: S.17.03.01.02, r: R0100, z: Z0001, filter: [s2c_dim:TB] = [s2c_LB:x28]}) <= {t: S.17.01.01.01, r: R0020} + {t: S.17.01.01.01, r: R0070} + {t: S.17.01.01.01, r: R0170}	Sum of column C0100 rows R0010, R0020, R0030 for table S.17.03.01.01 and column C0100 row R0100 for table S.17.03.01.02 for s2c_LB:x28 (Direct Business) is less than or equal to sum column C0100 of rows R0020, R0070 and R0170 for S.17.01.01.01

III.2.8 Exclusion of dictionary element (previously 'Member is not allowed')

This expression is used to indicate that a specific dictionary element cannot be used. This check is defined indifferently of the tables and aims to put the restriction on the entire report.

Validation	Explanation
{ m: [s2md_met:ei1904]} != [s2c_CU:x7]	Value s2c_CU:x7 (Temporary identifier for currency) should not be reported for metric ei1904 (Metric: Swap delivered currency (for buyer))
dim({d: [s2c_dim:LR]},[s2c_dim:LR]) != [s2c_GA:x112]	The item "Temporary identifier for country 1" must not be reported for dimension LR regardless of the table

III.2.9 Sum and maximum / minimum operators

Validation	Explanation
{t: SR.26.01.04.02, r: R0483, c: C0060, z: Z0001} = imax(0, ({t: SR.26.01.04.01, r: R0483, c: C0020, z: Z0001} - {t: SR.26.01.04.01, r: R0483, c: C0030, z: Z0001}) - ({t: SR.26.01.04.01, r: R0483, c: C0040, z: Z0001} - {t: SR.26.01.04.01, r: R0483, c: C0050, z: Z0001}))	Value reported in SR.26.01.04.02 row R0483 column C0060 must be equal to the maximum value of the two figures, zero and the difference between the result of subtraction of C0030 from C0020 and C0050 from C0040 reported for row R083 in SR.26.01.04.01 table
sum({t: S.14.01.01.05, c: C0075}) = {t: S.12.01.01.01, r: R0270, c: C0160}	Sum of values reported in S.14.01.01.05 column C0075 must be equal to value reported in S.12.01.01.01 row R0270 column C0160

III.2.10 Equivalence checks

Due to the data centricity of the model, equivalence checks are created between different datapoints. Creating a rule that checks the correspondence of reported values for the same datapoints misses the point and as such should be avoided for the sake of performance.

Validation	Explanation
scope({t: S.23.02.01.01, c:C0040, f: solvency, fv: solvency2}) {t: S.23.02.01.01, r: R0200, z: Z0001} = {t: S.23.02.01.01, r: R0120, z: Z0001}	Value reported in table S.23.02.01.01, row R0200 must be equal to the one reported in R0120 for column C0040 (Tier 2)
matches({t: S.06.02.07.02, c: C0290, z: Z0001, seq: False, id: v3, f: solvency, fv: solvency2}, "^..8.$")	The item "issuer code" in template S.06.02 - List of assets should be different from the item "identification code of the third country branch" from template S.01.02 - Basic Information - General for assets with CIC '##8#'

{t: S.06.02.07.02, c: C0210, z: Z0001} != {t: S.01.02.07.01, r: R0050, c: C0010}	