Release Notes

Visicon Smart BIM Tools

Version 2.2 ● September 10, 2021

Visicon Inc. – 340 E 77th St., NY, NY 10075 - +1 (347) 454-6229

support@visicon.com - visicon.com

Release 2.2

- Added support for Autodesk Revit 2022 export plugin.
- Added Left() and Right() expression functions.
- Added MatchesFormat function to Component Matching operation as a new expression-based matching option for strings with different formats. This new function overcomes situations where a user is trying to match component properties in two models, but the property in each model has been setup using a different format. This is commonly the case when trying to match steel members defined in Tekla vs the corresponding member in Revit.
- Fixed reported bugs.

MatchesFormat usage instructions:

Use the MatchesFormat function in the expression option of Component Matching to compare values of variables in Input A vs Input B. The function can be configured to detect specific formatting of each input string and compare sub-sets of characters. For example, it can compare the two strings W310x84 and 84*310 and check for the matching 310 and 84 characters even though the formatting of each variable is different.

Syntax: MatchesFormat(TxtA : String, FmtA : String, TxtB : String, FmtB : String)

Parameters:

- TxtA : Text A
- FmtA : Expected format for text A (See Format Specification below)
- TxtA : Text B
- FmtB : Expected format for text B (See Format Specification below)

The function returns a True value if the following 3 conditions succeed, and False otherwise.

1. TxtA matches FmtA
2. TxtB matches FmtB
3. Comparison of all values in A against all values in B is successful

Format Specification: the format of an input string is defined using any sequence of Text, Wildcards and Values.

Text

- Any text

Wildcards

- ? : Any single character

- * : Zero or more characters
- + : One or more characters
- Note: To use a wildcard character as a text literal, use two wildcards in a row. For example, to find strings that end with “5*5”, use the format “5**”

Values

Syntax: {[Index]:[Type][Length]i[Include]x[Exclude][@Compare Precision]}

- Index (Required) : any positive integer [0,inf]. This is used to match which values from FmtA and FmtB to compare.
- Type (Required):

Type	Description	Example
i	Integer	{0:i} = “-1”, “0”, “-34”, “63”, or “101”
iu	Integer (unsigned)	{0:u} = “1”, “0”, “34”, “63”, or “101”
f	Fraction with at least 2 numbers	{0:f} = “1/2”, “3 3/8”, “-5/6”, or “93 2/5”
fu	Fraction (unsigned)	1/2, 2, 3 3/8, 5/6, 93 2/5
d	Decimal	{0:d} = “-53.134”, “0”, “45”, “0.5”, “-.5”, or “144.135561”
du	Decimal (unsigned)	53.134, 0, 45, 144.135561
t(n|a)(s)	Text Options: 'n' for numeric text only, 'a' for alphabetic text only, and 's' to separate by whitespaces	{0:t} = "abc12 3!@_#$ ZX" {0:ts} = “the red fox” or “big dog” {0:tn} = “345” or “87823” {0:tns} = “34 58” or “4 879” {0:tns1-3} = “4 56” or “34 587” (Text of 1-3 characters length consisting of numeric characters separated by whitespace(s))
T(n|a)(s)	Text (case sensitive)	Same as [t] but with case-sensitive comparisons

- Length (Optional):

For non-fraction types:

Type	Description	Example

n	n characters	{0:t3} = "abc" or “1b%” {0:i3} = "123" or "-521" {0:iu} = “123” or “650”
n-m	n to m characters	{0:t1-3} = "a" or "ab" or "abc"
n+	n or more characters	{0:t2+} = "ab", "abc", "abcd" {0:i1+} = "3" or "-12345521"
n-	n or less characters	{0:t3-} = "a" or "ab" or "abc"

For decimal type only:

Type	Description	Example
n.p	n integral and p fractional characters	{0:d3.2} = "123.45", "169.23"
n-m.p-q	n to m integral and p to q fractional characters	{0:d1-3.2-3} = "123.45", "123.456"
n-.p	n or less integral and p fractional characters	{0:d3-.2} = "123.45" and "-1.23"
n+.p-	at least n integral and at most p fractional digits	{0:d3+.2-} = "123.45" and "12345" and "-12345.1"

For fractional types:

Type	Description	Example
1-2	A single whole number or a numerator and denominator	{0:f1-2} = "1" or "2/3"
1-3	A single whole number, a numerator	{0:f1-3} = "1" or "2/3" or "1 2/3"

	and denominator, or all three	
2	A numerator and denominator	{0:f2} = "2/3" (default, same as {0:f})
2-3	Whole number is optional, numerator and denominator must be present	{0:f2-3} = "2/3" or "1 2/3"
3	Whole, numerator, and denominator	{0:f3} = "1 2/3"

- Include (Optional, text types only):

Type	Description	Example
i[a-b]	Include characters a to b	{0:ti[a-z]} = {abcdefghijklmnopqrstuvwxyz} {0:ti[d-q]} = {defghijklmnopq} {0:ti[d-q]s} = “ef lm” or “fgh mo”
i[a,b,c]	Include characters a, b, and c	{0:ti[f,q,@]} = {fq@}
i[a,b,c-d]	Include characters a and b plus range c-d	{0:ti[w,r,A-Z]} = {wrABCDEFGHIJKLMNOPQRSTUVWXYZ} {0:t3i[a-z,A-Z]x[p,P,q,Q]} = “ftp” (Text of 3 characters length, consisting of characters "a-Z", excluding characters p,P,q,Q)

- Exclude (Optional, text types only):

Type	Description	Example
x[a-b]	Exclude characters a to b	{0:tx[a-z]} = {abcdefghijklmnopqrstuvwxyz}
x[a,b,c]	Exclude characters a, b, and c	{0:tx[f,q,@]} = {fq@}

		{0:ti[a-z,A-Z]x[p,P,q,Q]} = “afd” or “hop” (Text consisting of characters "a-Z", excluding characters p,P,q,Q)
xt[a,b,c-d]	Exclude characters a and b plus range c-d	{0:tx[w,r,A-Z]} = {wrABCDEFGHIJKLMNOPQRSTUVWXYZ}

- Compare precision for numeric types only (Optional):

Type	Description	Example
@4%	Compare numbers within 4%	{0:d@5%} = decimal within 5%
@5.23	Compare numbers within 5.23 internal units	{0:f@2.5} = any fraction within +/- 2.5 internal units

Examples of using MatchesFormat in Component Matching

The examples below are pulled from the use case where we want to compare and validate the steel fabrication data modeled by the fabricator in Tekla against our design intent model in Revit. We assume that relevant data for corresponding members in the Tekla and Revit models are found in the VarA and VarB parameters, respectively. Our goal is to define expressions that can automatically compare the values of these two parameters found in different models. Not only are the parameter names not consistent between the two models, but the formatting of their data is also inconsistent. The new MatchesFormat function lets us overcome this challenge and lets us automate the comparison of model data.

We would have had to merge the two models in a Visicon project before running these expressions and setup the Operation to compare one model as Input A and the other as Input B. Different potential values for each parameter VarA and VarB are also listed in the columns below.

Input A (VarA)	Input B (VarB)	Expression Example
W24X131 W8X58	W24X131 W8X58	a.var=b.description In this simple example, we are making a direct comparison between the two strings.

W24X131 W8X58	W24X131 W8X58	MatchesFormat(a.name, "?{0:i1-2}*", b.description, "?{0:i1-2}*") We are matching the name and description properties by skipping the first character using the ? wildcard symbol and then looking for an integer, assigned to index 0, that is between 1 and 2 characters in length. This expression succeeds in matching both the W24… and W8… paired components.
W24X131 W8X58	W24X131 W8X58	a.name.right(3) = b.description.right(3) This expression compares the 3 right-most characters of each input string.
W24X131 W8X58	W24X131 W8X58	a.name.left(1) = b.description.left(1) This time we are only comparing the left-most character of each input string. This could be used if you are just interested in checking whether you have matching W or C sections, for example, but don't care about the details of the individual members.
HSS6x6x.375 HSS6x6x.999	HSS6x6x3/8 HSS6x6x3/8	MatchesFormat(a.VarA, "HSS*{0:d@2%}", b.VarB, "HSS*{0:f}") This function is configured to check the decimal value found in VarA after the text HSS and any number of characters against the fraction found after a similar configuration of characters in VarB. Visicon parses the input strings and isolates different sub-strings based on their type classification. Both values are assigned to index 0 and will be compared using a 2% tolerance against the decimal value.
HSS6x6x.375 HSS6x6x.999	HSS6x6x3/8 HSS6x6x3/8	MatchesFormat(a.VarA, "HSS6x6x{0:d3-.3-@2%}", b.VarB, "HSS6x6x{0:f}") This version of the expression tests for the same 2% compatibility between the decimal value in Vara and the fraction in VarB with the difference that it does not use a wildcard and defines the other characters explicitly.

310UC158 310UC158	UC310*158 UC310*154	MatchesFormat(a.VarA, "{0:t3}*{1:t3}", b.VarB, "??{0:t3}*{1:t3}") This expression is setup to test for two values within each string: one defined as index 0 and the other as index 1. VarA is setup to assign the first three characters to index value 0 and the last three characters to index value 1. The string can have any number of characters in between, represented by the wildcard *. VarB allows for any 2 characters and then assigns the next three to index value 0 and the last three to index value 1. These two sets of characters can be separated by any number of other characters.
W24X131 W8X58	W24X131 W8X58	MatchesFormat(A.VarA, "?{0:i1-2}*", B.VarB, "?{0:i1-2}*") The matching algorithm works through the input strings from the left to the right, parsing the strings based on the formatting parameters and testing to see if the values for each index value match. In this case, each input string's first character is skipped and then the next set of one to two characters are matched. This flexible definition lets you compare sub-strings that may vary in length.
W123 x 40 x 50 W123 x 50 x 50	W123 x 40 x 50 W123 x 40 x 50	MatchesFormat(a.VarA, "W123 x {1:t1-3} x 50", b.VarB, "W123 x {1:t1-3} x 50") This expression only compares the middle values. These could be 1 to 3 characters in length.
W123 x 40 x 50 W123 x 50 x 50	W123 x 40 x 50 W123 x 40 x 50	MatchesFormat(a.VarA, "W{0:t1-3} x {1:t1-3} x {2:t1-3}", b.VarB, "W{0:t1-3} x {1:t1-3} x {2:t1-3}") Here we have setup the expression to check all three sets of sub-values independently. Each sub-value is assigned to its own index and can be 1 to 3 characters long.
W123 x 40 x 50 W123 x 50 x 50	W123 x 40 x 50 W123 x 40 x 50	MatchesFormat(a.VarA, "W{0:t3}???{1:t2}???{2:t2}", b.VarB, "W{0:t3}???{1:t2}???{2:t2}") This version of the expression tests the same 3 sub-values as the previous example, but uses wildcard characters for the filler text.

.375 .500	3/8 3/8	MatchesFormat(a.VarA, "{0:d3-.3@2%}", b.VarB, "{0:f}") We are comparing a decimal value with up to 3 integral and three decimal characters against a fraction with a 2% tolerance.