Release History

Visicon Smart BIM Tools

Updated September 10, 2021

Visicon Inc. – 340 E 77th St., NY, NY 10075

+1 (347) 454-6229 - support@visicon.com - visicon.com

Release 1.0

Initial release.

Release 1.1

- Corrected known bugs.
- Improved Model Comparison operation.
    - Added XLSX export option to Model Comparison operation.
    - Added Import with Issus to Model Comparison operation.
    - Improved Model Comparison object matching and reporting in UI.
- Enhanced capabilities for model merging, updating and alignment.
    - Now support merging and updating of VXF files containing multiple models.
    - Added model-specific options to merge, update and align.
    - Store manual alignment data and give option to apply to newly imported models when updating.
    - Give new options to apply model alignment when updating models.
    - Simplified alignment widget.
- This version now creates separate models within Visicon when exporting linked models from Revit.
- Extended tendon modeling logic in Visicon when importing INP files.
    - Added tendon end color properties.
    - Added tendon end angle properties and adjust anchor splay using these values.
    - Automatically select bonded vs unbonded tendon system.
- Added the User Guide and Getting Started Guide to the Windows start menu.
- Relaxed reliance on AVX instructions to support older CPUs.
- Added more details to clash and clearance operation issues.
- Added option to define custom variables for Inventory feature.
- Refined ETABS E2K file support.
    - Support greater number of frame section types.
    - Reason about cardinal insertion point for all component types.
- Added cancel button for computationally intensive Operations.

Release 1.2

- Corrected known bugs
- Added Revit 2020 support
- Implemented improved Revit Plugin for versions (2015 – 2020)
    - Simplified UI
    - Added immediate loading to view export options
    - Support for user-editable export profiles
    - Can now export components on specific view only
- Report Top Level assignment when using Spot Elevation on component
- News dialog now opens at startup when new news available
- Improved component compression in VXF file
- Added basic/advanced shading option
- Provided new navigation option that emulates Navisworks Walk
- Added walk movement and walk rotation speed settings
- Rectangular Select navigation mode now emulates Revit navigation style
- Allow alignment without using widget by selecting components only
- Added round(x) and nearest(x,y) expression functions to allow variable rounding in reports and expressions
- Now support ability to add new custom parameters to components in model
    - Add new parameters using right-click Insert option under Object Properties Parameters/Family tabs or Project Browser Family panel
    - Define units of new parameter
    - Automatically calculate value of new parameter using an Expression
    - Use Insert Variable option to select from a list of available parameters and expression functions
- Project Browser Parameters panel has new unit column
- Added “Modified Only” variance option for Variance Reports and Operations
- Each model in a Visicon project now has an automatic selection filter in the reports and operations filter options pull-down, eliminating the need to write separate filters
- Added component-type Boolean Difference mode for cases where component IDs may not match
- Added new Low Memory Mode option in Project Browser and Revit Plugin.
- New Save as External option was added to File menu
    - Visicon components can now be exported in OBJ generic 3D object file format for further re-processing and import into other programs
    - File menu option saves OBJ as world-space meshes
    - Right-click option saves OBJ as local space mesh
    - File menu save as OBJ option will save selection or entire project if no selection

Release 1.3

- Corrected known bugs
- Implemented object snapping option that can be used with markups and measurement – each can be further refined to focus on points, edges, intersections and angles
    - Surface
    - Centerline
    - Gridline
    - Clip Place Sections
- Enhanced 2D markups with free-form Pen function
- Added new fonts for text markups
- Simplified Revit plugin user interface
- Added resolution setting to Save Screenshot function
- Improved report formatting for XLS and PDF formats
- Added ability to restrict Inventory (Quantity) Reports to components visibly shown while using clip planes. Select option to “Consider Clip Planes” = Yes.
- Added Table of Contents Report
- Added Project Information Report
- Added Component Matching report that lists column attributes for two different models that are expected to represent the same project
- Extended measurement features by adding volume and count options
- Introduced flexibility of letting user force 3D markups and measurements to always be on top and not scale
- Variable inputs can now support optional tolerance setting, e.g., “Volume @ 2%”
- Variables inputs can now support optional precision parameter with units, e.g., “Length @ 3mm” or “Length @ 0.01”
- Created new 3D markup Marker object type
- Improved Clash Check Operation
    - Option to include Contact Tolerance, Penetration Tolerance, and Penetration Tolerance at Extremity
    - New option to auto-group issues associated with the same affected component
- Improved Boolean Difference Operation by adding options to detect Added, Removed, Moved and Changed geometries
- USB-based licenses are now locked to the individual USB key and can be moved from computer to computer without requiring a new activation
- To support faster setup of trial licenses, activation requests can now be requested without the need for a purchase code

- New News feature is now active and will initiate a new interface with up-to-date news whenever new content is made available by Visicon
- Variance Operation and Report now support ability to check for multiple variables at one time – quantity and parameters
- Added “=expression” support for component properties
- Added ability to open and import any Visicon compatible file (except for point clouds) into Revit as a generic 3D mesh object
- New dynamic Revit component selection option from within Visicon
- Extended ETABS E2K support to read a more complete set of non-geometric component parameters

Release 1.3.1

- Corrected known bugs
- Extended capabilities to support special language characters in:
  - HTML and PDF reports
  - Expressions
  - Revit export
- Improved parameter naming when encountering multiple, similar parameters
- Improved compression algorithm
- Increased number of decimal places displayed for measurements

Release 2.0

- Corrected known bugs
- Introduced new Standard product tier
- Revit 2021 compatibility
- Extended support for 100% of ETABS section types
- Automatic 30-day trial option
- New, flexible licensing options
    - Cloud network license
    - Self-hosted network license
    - Online system locked license activation
- Improved zooming behavior for close-up conditions
- Improved navigation performance for when accessing Visicon through remote desktop
- Simplified user interface by removing unused collaboration ribbon
- Added “” to expression syntax – used to represent “ within text variables

Release 2.0.1

- Added new Component Matching operation
- Improved mouse scrolling speed when in Free Navigation mode
- Added new Function **Equals(a,b,tolerance)** for expressions that allows user to specify an equivalency test for a component variable within a given tolerance – tolerance is specified in units of variable.
    - Example usage: Equals(Length,5ft,1)
        - The above expression checks for lengths within 1ft of 5ft
- Added new Function **Equalspercent (a,b,tolerance)** for expressions
    - Example usage: Equalspercent(Length,8ft,20)
        - The above expression checks for lengths within 20% of 8ft
- Added **Approx Equals Tolerance %** for Expressions in Checking Rules
    - Example usage: Length ~= 10ft

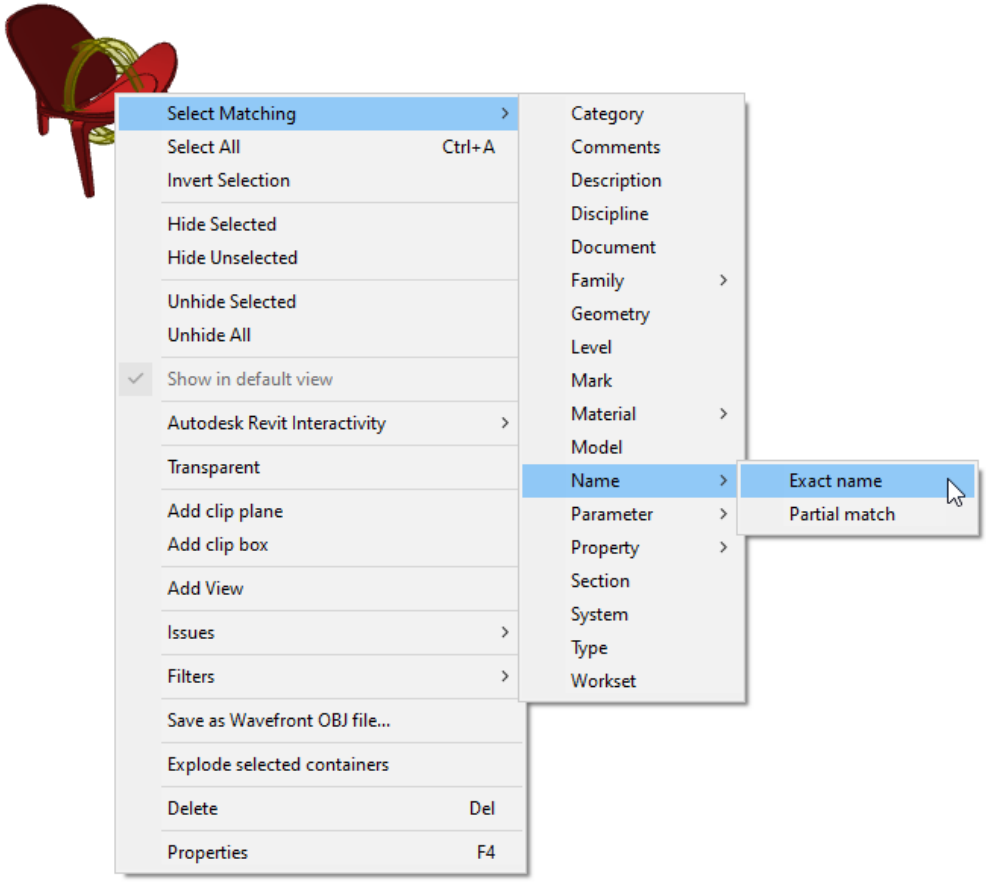
Release 2.1

- Corrected known bugs
- Overhauled navigation to give user better and more intuitive options
    - Previous Free option is now set at Default mode when opening new models
    - In Default navigation mode, user can switch rotation point by selecting any component
    - Zoom is directional and centers in on the cursor's location
    - Previous Orbit and Walk mode navigation options are retained
- Extended clip plane manipulation to include rotation and translation of widget
- Added new Clip Box feature to isolate selected components using 6 automatically generated clip planes – invoke using right-click mouse option
- Added F1 Help option to icons in Revit plugin
- Improved Component Matching Operation
    - New Center Tolerance option for comparing components with irregular geometries
        - Paired components are matched at a mid-height section
        - The original Centroid Tolerance option calculated the actual centroid of a components' section
        - The Center is calculated as the center of the bounding box (aligned with global X and Y coordinates) of the components' cross section
    - Added section approximation for irregular shapes
- Refined Boolean Difference Operation to explicitly create Added/Removed sections for all changes, not just those that were true Boolean geometries
- Enabled alignment snapping option when Manually moving components or models
- Added new global Functions that help match strings from different components in the Component Matching Operation
    - **ReplaceText(Name, "a", "b")** replaces the "a" string with "b" string in the component variable Name
        - Example usage: ReplaceText(Description, "*", "x")
        - This would switch the Description text "W14*122" to "W14x122"
    - **UpperText(Name)** switches all characters in the Name component variable to upper case
        - Example usage: UpperText(Name)
        - This would switch the Name text "Column 1" to "COLUMN 1"
    - **LowerText(Name)** switches all characters in the Name component variable to lower case
        - Example usage: LowerText(Name)
        - This would switch the Name text "Column 1" to "column 1"

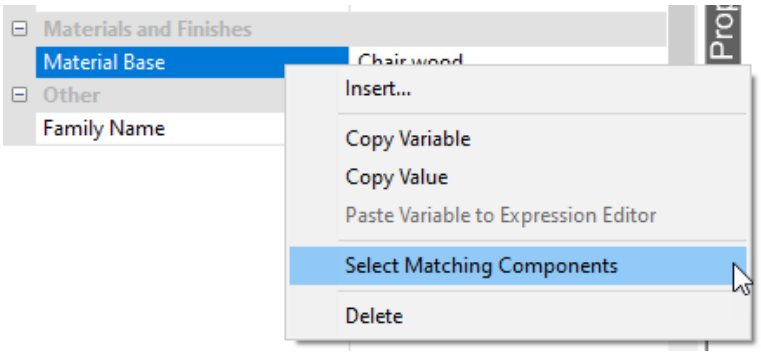
- Added new variable Functions that help match strings from different components in the Component Matching Operation
    - **Name.Replace("a", "b")** replaces the "a" string with "b" string in the component variable Name
        - Example usage: Description.Replace("*", "x")
        - This would switch the Description text "W14*122" to "W14x122"
    - **Name.MakeUpper()** switches all characters in the Name component variable to upper case
        - Example usage: Name.MakeUpper()
        - This would switch the Name text "Column 1" to "COLUMN 1"
    - **Name.MakeLower()** switches all characters in the Name component variable to lower case
        - Example usage: Name.MakeLower()
        - This would switch the Name text "Column 1" to "column 1"
- Implemented silent installation option that system administrators can use to automate the rollout of Visicon to end-users – contact info@visicon.com for instructions on how to utilize this capability

Release 2.1.1

- Corrected 2 ETABS (E2K) related bugs
  - In rare cases, braces were placed at the wrong level
  - For specific beams, the material assignment was wrongly mapped
- Increased timeout duration for users that are using Visicon's cloud-based network licenses
- Extended Select Matching right-click mouse option for selected components to include a wider range of options



- Added right-click mouse option to all object properties and parameters to allow user to Select Matching Components – this new feature lets users quickly select other components in the model that have the same property value.



Release 2.1.2

- Corrected an ETABS (E2K) related bug where models created by the wizard were not being imported properly.
- Fixed a clash detection bug related to components with very irregular geometry.

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.