

MultiPushTool: Pattern Parser (PP) Dokumentation

[MultiPushTool: Pattern Parser \(PP\) Dokumentation](#)

[Bekannte Probleme:](#)

[Syntax Revit Parameter](#)

[Syntax Erweiterte Funktionen](#)

[Syntaxvereinheitlichung ab Version 2016](#)

[bimmFloorParameter](#)

[Boundingbox](#)

[FirstMaterial](#)

[GeometryFilterParameter](#)

[QTO](#)

[LevelParameter](#)

[BaseLevelParameter](#)

[TopLevelParameter](#)

[RoomParameter](#)

[FromRoomParameter](#)

[ToRoomParameter](#)

[HostParameter](#)

[PartHostParameter](#)

[Perimeter](#)

[Today](#)

[Syntax Textfunktionen](#)

[Verketteten von Texten \(auch aus Parametern\)](#)

[Concat\("TEXT1", ..., "TEXT4"\)](#)

[Contains\("TEXT"\)](#)

[EndsWith\("TEXT"\)](#)

[IndexOf\(Zeichenfolge\)](#)

[Insert\(StartWert"TEXT"\)](#)

[IsNullOrEmpty\(Zeichenfolge\)](#)

[IsNullOrWhiteSpace\(Zeichenfolge\)](#)

[Length \(Vorsicht: hier ohne Klammerpaar am Ende\)](#)

[Remove\(StartWert\)](#)

[Remove\(StartWert, Anzahl\)](#)

[Replace\("AlterText", "NeuerText"\)](#)

[StartsWith\("TEXT"\)](#)

[Substring\(StartWert\)](#)

[Substring\(StartWert, Länge\)](#)

[ToLower\(\)](#)

[ToString\(Formatierung\)](#)

[ToUpper\(\)](#)

[Konstanten](#)

[Datum- und Zeitangaben](#)

[Now](#)

[Konvertierung](#)

[ToDateTime\(Zeichenfolge\)](#)

[ToDouble\(Zeichenfolge\)](#)

[ToInt32\(Zeichenfolge\)](#)

[ToString\(Zahl\)](#)

[Bedingungen](#)

[Beispiele:](#)

[Trigonometrische Funktionen \(->Link zu Wikipedia\)](#)

[Sonstiges](#)

[Setzen von Ja/Nein Parametern in Revit](#)

Bekannte Probleme:

- **Ganzzahl Division funktioniert momentan nicht korrekt!!! Es wird nicht korrekt gerundet!**

Beispiel: $1 / 2 = 0!$ oder $5 / 2 = 2!$

Abhilfe: Sobald eine Zahl eine Gleitkommazahl ist, wird richtig gerechnet

- Wenn man Regeln über den **Button "Datei speichern"** in eine externe mpxml Datei speichern und eine vorhandene mpxml Datei überschreiben möchte, öffnet sich im Hintergrund ein Warndialog, leider für den Benutzer nicht sichtbar. Dieser scheint dann das gesamte Programm zu blockieren.

Abhilfe: Durch einfaches drücken der **"ALT" - Taste** wird der Warndialog in den Vordergrund gesetzt und somit sichtbar für den Benutzer

WICHTIG:

- Berechnungen, Ausdrücke und Funktionen immer innerhalb eines geschweiften Klammerpaars **{...}**
- Dezimaltrennzeichen ist der Punkt ".", also **2.7** (nicht 2,7)

Syntax Revit Parameter

Die Bibliothek bietet Zugriff auf beinahe alle Parameter der Revit Datenbank

Allgemeiner Syntax:

[**TOKENTYP:ParameterName**]

Projekt(Information)parameter (Tokentyp: PROJECT)

Z.B.: [PROJECT:Projektname]

Exemplarparameter (Tokentyp: INSTANCE)

Z.B.: [INSTANCE:Kennzeichen]

Typparameter (Tokentyp: TYPE)

Z.B.: [TYPE:Breite]

Exemplar-Builtin-Parameter (Tokentyp: BIPINSTANCE)

Z.B.: [BIPINSTANCE:Id]

Typ-Builtin-Parameter (Tokentyp: BIPTYPE)

Z.B.: [BIPTYPE:Gesperrt]

Ab Version 2015.1.0.0:

Eigene Auswahllisten (Tokentyp: SHORTLIST)

Z.B.: [SHORTLIST:NUTS]

Erweiterte Funktionen

Tokentyp: FUNCTION

Syntaxvereinheitlichung ab Version 2016

Einige erweiterte Funktion haben Argumente in Form von Revit-Parametern. Bis Version 2015 wurden alle Argumente durch Kommata getrennt. Tokentyp und Tokenname werden ab Version 2016 immer durch Doppelpunkt getrennt.

Z.B. [FUNCTION:LevelParameter(INSTANCE:Name)]

Im Unterschied zu Parameter-Tokens werden hier keine eckigen Klammern verwendet.

Die Dialogfelder der jeweiligen Funktionen geben die richtige Syntax aus. Die alte Syntax kann aber weiterhin verwendet werden.

Übersicht der erweiterten Funktionen:

- [ArchitecturalFloorParameter](#)
- [BaseArea](#)
- [bimmFloorParameter](#)
- [bimmStoreyModelInfo](#)
- [BoundingBox](#)
- [CeilingParameter](#)
- [FinishRoomHeight](#)
- [FirstFloorParameter](#)
- [FirstMaterial](#)
- [FloorAboveParameter](#)
- [FromRoomParameter](#)
- [GeometryFilterParameter](#)
- GetAreaOfOpenings
- [HostParameter](#)
- [LevelParameter](#)
- [PartHostParameter](#)
- [Perimeter](#)
- ProjectCalculation
- ProjectSumUp
- [QTO](#)
- [RoomParameter](#)
- [RoughRoomHeight](#)
- [StructuralFloorParameter](#)
- SumUp
- TitleBlockParameter
- [Today](#)
- TopArea
- [TopLevelParameter](#)
- [ToRoomParameter](#)

ArchitecturalFloorParameter

Ab Version 2017.1.3.0

Funktion für Kategorie Räume

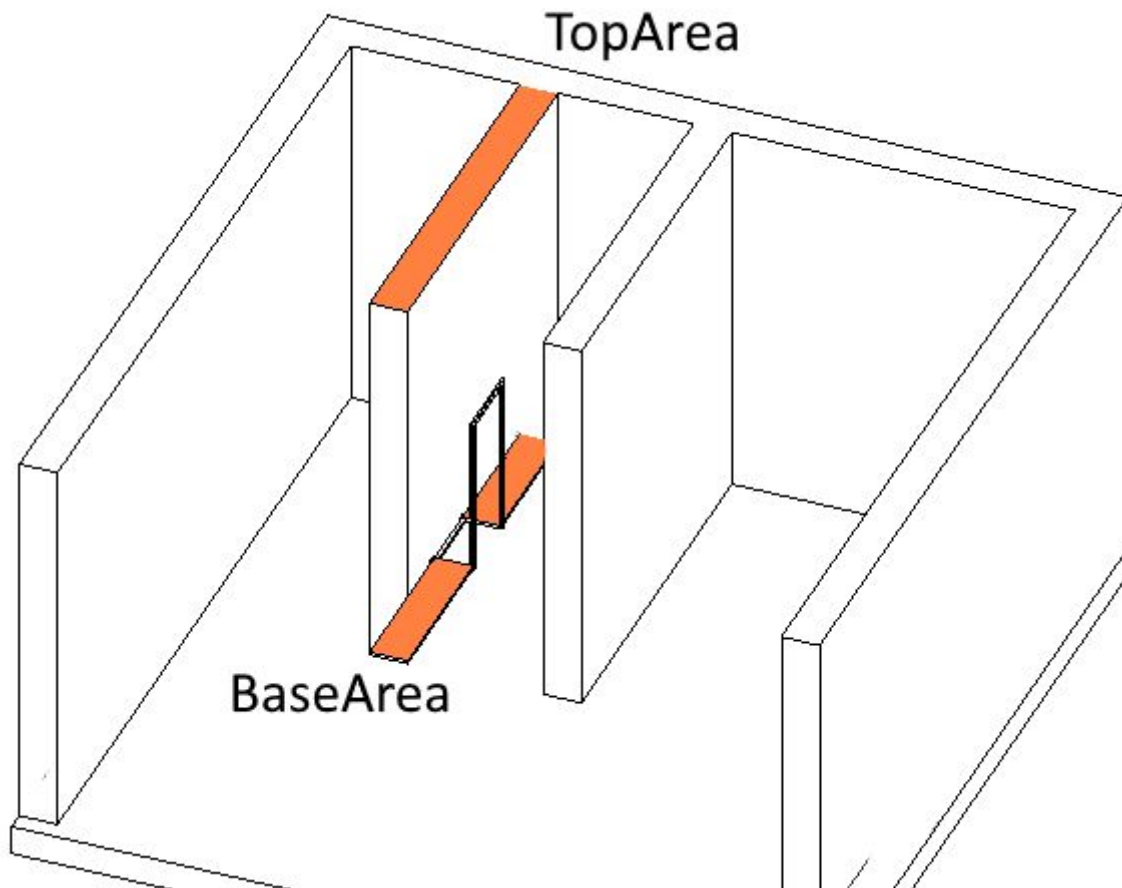
Bietet Zugriff auf die Parameter des zugehörigen Fußbodenelements (nichttragende Geschossdecke)

BaseArea

Ab Version 2017.1.3.0

Funktion für Kategorien Wände & Stützen

Ermittelt die Standfläche (Konstruktionsfläche) von Wänden und Stützen unten



Siehe auch TopArea-Funktion

BaseLevelParameter

(Ab Version 201X.X.X.0)

(noch zu implementieren)

bimmFloorParameter

Funktion für Kategorie *Räume*

Bietet Zugriff auf die Parameter des zugehörigen Fußbodenelements (über RoomFloorTool generierte FB-Geschoßdecke)

bimmStoreyModelInfo

Bietet Zugriff auf alle Parameter des b.i.m.m Geschossmodells (über LevelManager erstellt)

Select Storey Attribute

Name

StoreyUsageCode_StoreyFormattedNum

Connector: _ Preview: OG_01

Insert Parameter

Filter...

Name	Value
StoreyFormattedNum	01
StoreyUsageName	Obergeschoss
StoreyUsageCode	OG
StoreyUsageComment	
StoreyLocationName	oberirdisch
StoreyLocationCode	
StoreyLocationComment	
StoreyCode	1
StoreyModelCode	DE
LevelUsageName	OKFFB
LevelUsageCode	OKFFB
LevelUsageComment	
LevelComment	
LevelElevation	10,259186351706
LevelComponent	

<

>

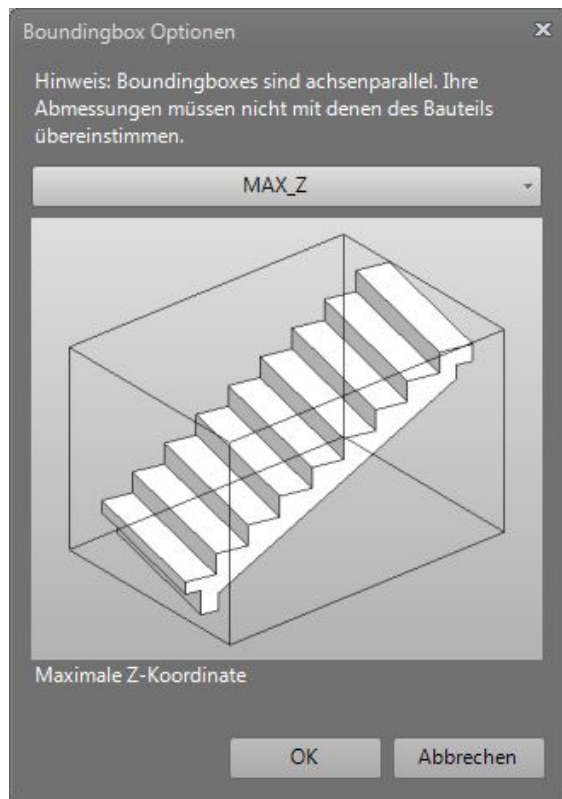
OK

Cancel

BoundingBox

Für alle Modellgeometrie-Kategorien

Bietet Zugriff auf Geometriedaten der Boundingbox eines Elements



Folgende Werte können über diese Funktion in die ausgewählten Elemente geschrieben werden:

- ☐ MAX_X ; MIN_X
- ☐ MAX_Y ; MIN_Y
- ☐ MAX_Z ; MIN_Z
- ☐ LENGTH
- ☐ WIDTH
- ☐ HEIGHT
- ☐ CENTER_X ; CENTER_Y , CENTER_Z

Hinweis: Die Boundingbox eines Geometrie-Elements ist achsenparallel zu den Koordinatenachsen. Die Abmessung muss nicht mit denen des Bauteils übereinstimmen.

Bei fixierten Bauteilen wird die Höhe nicht korrekt ermittelt (Revit Bug)

CeilingParameter

Ab Version 2017.1.3.0

Für Kategorie Räume

Bietet Zugriff auf die Parameter der abgehängten Geschossdecke innerhalb eines Raumes, wenn vorhanden (Prüfung findet am Raumeinfügapunkt statt)

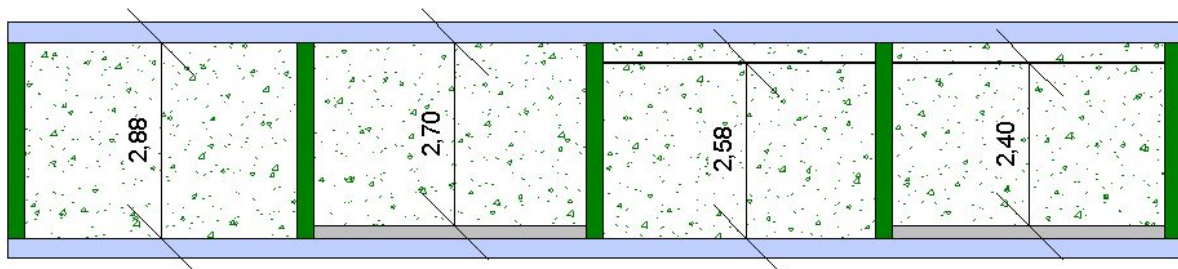
FinishRoomHeight

Ab Version 2017.1.3.0

Für Kategorie Räume

Ermittelt die lichte Raumhöhe am Raumeinfügepunkt

(Funktion berücksichtigt Tragende und Nicht-tragende Geschossdecken sowie abgehängte Decken)



Siehe auch RoughRoomHeight-Funktion

FirstFloorParameter

Ab Version 2017.1.3.0

Für Kategorie Räume

Bietet Zugriff auf alle Parameter der zuerst gefundenen Geschossdecke unterhalb des Raums, tragend oder nicht-tragend.

FirstMaterial

Bietet Zugriff auf die Parameter des ersten Materials eines Elements (Nur für einschichtige Bauteile gedacht)

FloorAboveParameter

Ab Version 2017.1.3.0

Für Kategorie Räume

Bietet Zugriff auf alle Parameter der zuerst gefundenen Geschossdecke oberhalb des Raums

FromRoomParameter

Funktion für ladbare Familien

Bietet Zugriff auf die "FromRoom"-Parameter.

GeometryFilterParameter

Bietet Zugriff auf die Parameter des verwendeten GeometryFilters.

HostParameter

Funktion nur für Familien, die Host-basierend sind (z.B. Türen: Host=Wand, in der sich die Türe befindet)

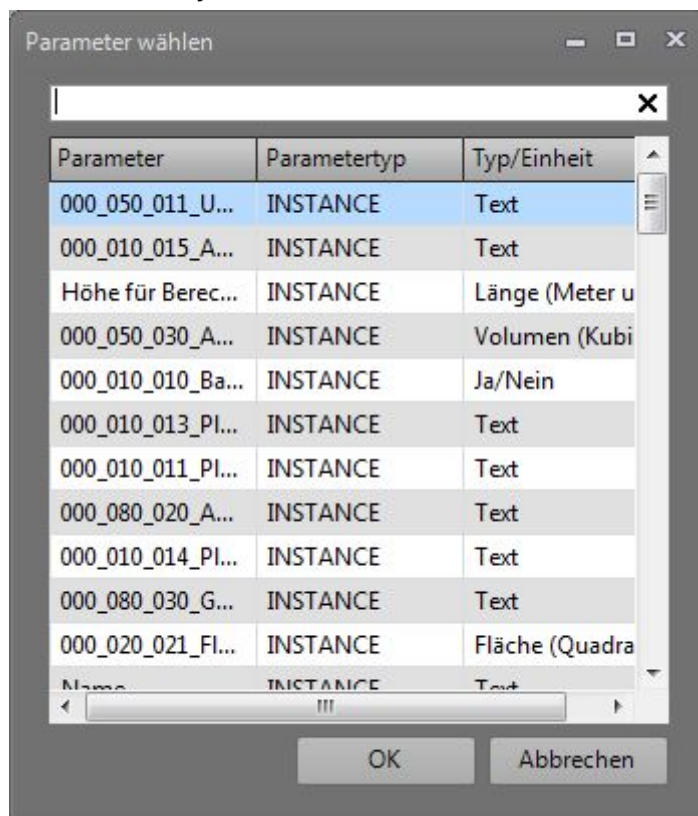
Bietet Zugriff auf die Parameter des Host-Elements (bei Türen z.B. die jeweiligen Wandparameter)

Siehe auch PartHostParameter-Funktion

LevelParameter

Zugriff auf die Parameter der Standardebene des Bauteils

Hinweis: Nicht jedes Bauteil hat eine Standardebene



Weiteren Text

Syntax:

[FUNCTION:LevelParameter(ARGUMENTE)]

Beispiel:

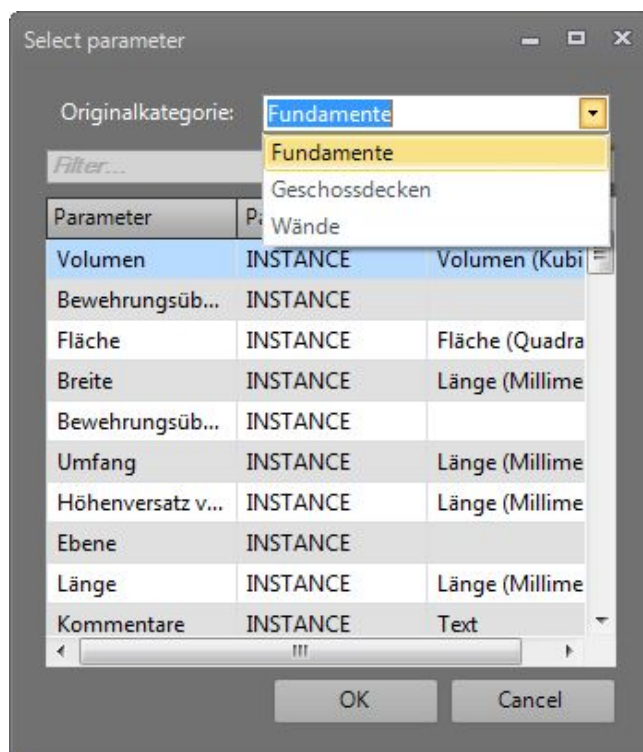
[FUNCTION:LevelParameter(INSTANCE:Ansicht)]

PartHostParameter

(Ab Version 2014.2.X.0)

Funktion nur für Kategorie *Teilelemente* (Parts)

Bietet Zugriff auf die Parameter des PartHost-Elements (d.h. Parameter der Originalfamilie z.B. die jeweiligen Wandparameter)



Syntax:

[FUNCTION:PartHostParameter(ParameterTypToken:ParameterName)]

ParameterTypToken:

- INSTANCE (Exemplarparameter; in der Revit Benutzeroberfläche sichtbar)
- TYPE (Typparameter; in der Revit Benutzeroberfläche sichtbar)
- BIPINSTANCE (Exemplarparameter; in der Revit Benutzeroberfläche NICHT sichtbar)
- BIPTYPE (Typparameter; in der Revit Benutzeroberfläche NICHT sichtbar)

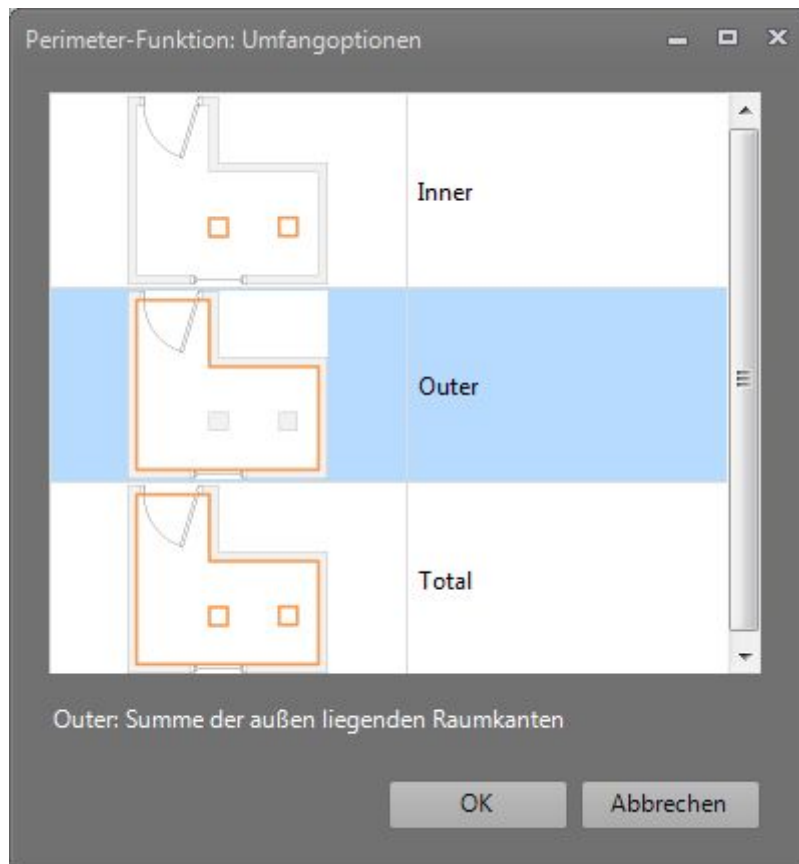
Beispiel:

[FUNCTION:PartHostParameter(TYPE:Typenmarkierung)]

{if([INSTANCE:Originalkategorie]="Wände",[FUNCTION:PartHostParameter(INSTANCE: Kennzeichen)],[CONSTANT:Ignore])}

Perimeter

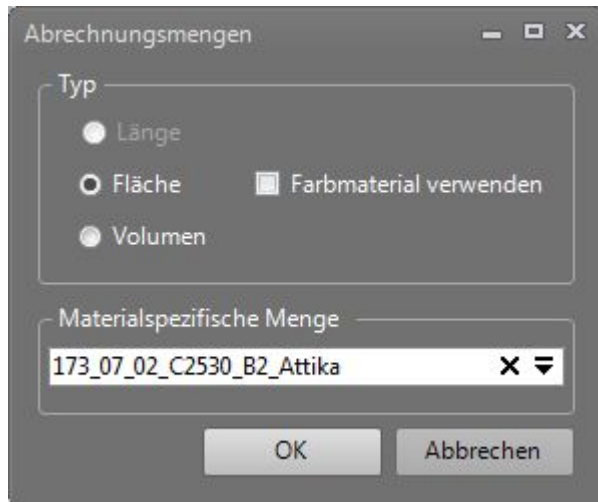
Funktion nur für Räume



Berechnet den äußeren, inneren und den Gesamtumfang eines Raums

QTO

Mit QTO können materialabhängige Massen ermittelt werden; die Funktion hat einen eigenen Unterdialog zur Auswahl der gewünschten Menge (Fläche oder Volumen) und des gewünschten Materials.



Wenn nur ein Rückgabewert gewünscht ist (direktes Hineinschreiben in einen Zahlenparameter) werden keine geschweiften Klammern benötigt; wird mit ihr weiter gerechnet bzw. ist sie Teil einer Bedingung, muß die Funktion sich innerhalb eines geschweiften Klammerpaares befinden.

Syntax:

[FUNCTION:QTO(ARGUMENTE)]

Beispiel:

[FUNCTION:QTO(AREA, 054_050_010_Treppenuntersicht)]
 {[FUNCTION:QTO(AREA, 054_050_010_Treppenuntersicht)]*1.05}

RoomParameter

(Funktion für ladbare Familien)

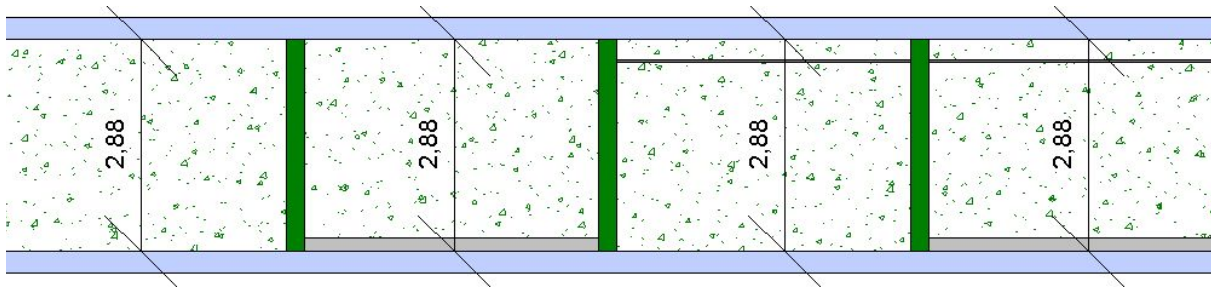
RoughRoomHeight

Ab Version 2017.1.3.0

Für Kategorie Räume

Ermittelt die Rohbauraumhöhe am Raumeinfügapunkt

(Funktion berücksichtigt ausschließlich tragende Geschossdecken)



Siehe auch FinishRoomHeight-Funktion

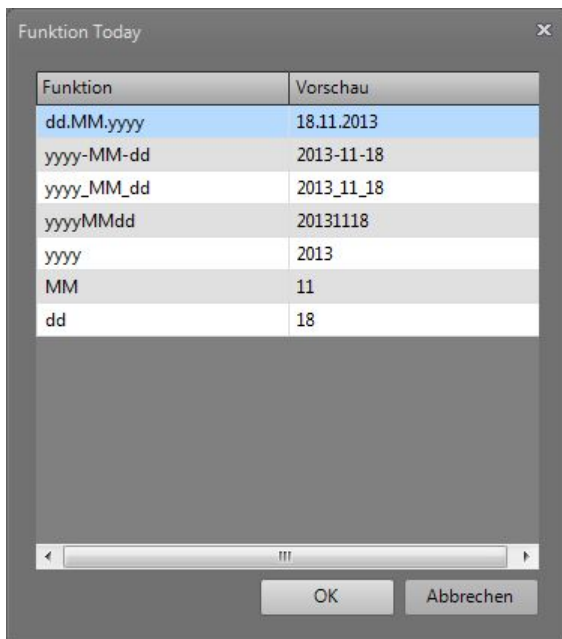
StructuralFloorParameter

Ab Version 2017.1.3.0

Für Kategorie Räume

Bietet Zugriff auf alle Parameter der tragenden Geschossdecke unterhalb des Raums.

Today



Einfache Hilfsfunktion zum Übertragen des aktuellen Datums in verschiedenen Formatierungen.

TopLevelParameter

(Ab Version 201X.X.X.0)

(noch zu implementieren)

ToRoomParameter

(Funktion für ladbare Familien)

Syntax Textfunktionen

Nachfolgend sind die wichtigsten Textfunktionen abgebildet, die unsere Bibliothek unterstützt. Parameter und Funktion werden mit einem Punkt "." verbunden, Text in Argumenten immer in "Anführungszeichen", zum Beispiel

```
{[INSTANCE:Kommentare].Insert(0, "b.i.m.m")}
```

Verketten von Texten (auch aus Parametern)

Beim Verketten von Texten werden folgende Fälle unterschieden:

- Ohne Formel: einfache Syntax, ohne Anführungszeichen etc. (siehe Beispiel)
- Innerhalb von Formeln bzw. Bedingungen {...}: Verketten von Texten wird mit dem Plus-Zeichen gemacht (+)

Beispiel:

- Ohne Formel: Text [INSTANCE:Kommentare]
- Innerhalb Formeln {...}:
{...[BIPINSTANCE:Typ].contains("Text"+[INSTANCE:Kommentare])...}

Concat("TEXT1", ... , "TEXT4")

(Ab Version 2014.2.0.0)

Verkettet maximal vier Zeichenfolgen (Text1 bis Text4)

Ergebnis: Neuer Text

Beispiel:

```
{Concat([BIPTYPE:Familienname], "-", [BIPTYPE:Typname])}
```

Contains("TEXT")

Gibt an, ob die gegebene Zeichenkette in dieser Zeichenkette vorkommt.

Ergebnis: Wahr / Falsch

Beispiel:

```
{[BIPINSTANCE:Typ].contains("_FASS_")}
```

EndsWith("TEXT")

Gibt an, ob die gegebene Zeichenkette das Ende dieser Zeichenkette darstellt.

Ergebnis: Wahr / Falsch

IndexOf(Zeichenfolge)

Gibt den NULL-basierten Index des ersten Vorkommens der angegebenen Zeichenfolge im verwendeten Parameterwert an.

Ergebnis: Ganzzahl

Beispiel:

```
{{INSTANCE:Kommentare}.IndexOf("_")}
```

Kommentare: "2,70_RDOK": ergibt 4 (Startwert 0)

Insert(StartWert"TEXT")

Einfügen eines Teilstrings

Ergebnis: Neuer Text

Beispiel:

```
{{INSTANCE:Kommentare}.Insert(0, "b.i.m.m")}
```

Fügt b.i.m.m gleich am Anfang des alten Werts ein (Startwert beginnt mit 0)

IsNullOrEmpty(Zeichenfolge)

(Ab Version 2014.2.0.0)

Prüft, ob die angegebene Zeichenfolge leer ist

Ergebnis: Wahr / Falsch

Beispiel:

```
{if(IsNullOrEmpty([INSTANCE:000_080_030_Geschoß]),"Kein Geschoss  
vorhanden",[INSTANCE:000_080_030_Geschoß])}
```

IsNullOrWhiteSpace(Zeichenfolge)

(Ab Version 2014.2.0.0)

Prüft, ob die angegebene Zeichenfolge leer ist, bzw. nur Leerzeichen beinhaltet

Ergebnis: Wahr / Falsch

Beispiel:

```
{if(IsNullOrWhiteSpace([INSTANCE:000_080_030_Geschoß]),"Kein Geschoss  
vorhanden",[INSTANCE:000_080_030_Geschoß])}
```

Length (Vorsicht: hier ohne Klammerpaar am Ende)

Ruft die Anzahl der Zeichen im aktuellen String-Objekt ab.

Ergebnis: Ganzzahl

Beispiel:

```
{[BIPINSTANCE:Typ].length
```

Ergibt 24 (Typname: 173_WT_A_BEWA_0200_C2530)

Remove(StartWert)

Löscht alle Zeichen aus dieser Zeichenfolge, beginnend an einer angegebenen Position bis zur letzten Position.

(Startwert beginnt mit 0)

Ergebnis: Neuer Text

b.i.m.m

Remove(1) ="b"

Remove(StartWert, Anzahl)

Löscht die angegebene Anzahl von Zeichen ab der angegebenen Position.

(Startwert beginnt mit 0)

Ergebnis: Neuer Text

b.i.m.m

Remove(1,1) ="bi.m.m"

Replace("AlterText", "NeuerText")

Ersetzen von Zeichenfolge durch gegebene Zeichenfolge.

Ergebnis: Neuer Text

Beispiel:

```
{[INSTANCE:Kommentare].Replace("bimm", "b.i.m.m-")}
```

Aus bimmKommentar wird b.i.m.m-Kommentar

StartsWith("TEXT")

Gibt an, ob die gegebene Zeichenkette den Anfang dieser Zeichenkette darstellt.

Ergebnis: Wahr / Falsch

Substring(StartWert)

Ruft eine Teilzeichenfolge ab. Die Teilzeichenfolge beginnt an einer angegebenen Zeichenposition (Startwert beginnt mit 0).

Ergebnis: Neuer Text

Beispiel:

```
{[BIPINSTANCE:Typ].substring(19)}
```

Ergibt C2530 (Typname: 173_WT_A_BEWA_0200_C2530)

Substring(StartWert, Länge)

Ruft eine Teilzeichenfolge ab. Die Teilzeichenfolge beginnt an einer angegebenen Zeichenposition und hat eine angegebene Länge. (Startwert beginnt mit 0).

Ergebnis: Neuer Text

Beispiel:

```
{[BIPINSTANCE:Typ].substring(4,2)}
```

Ergibt WT (Typname: 173_WT_A_BEWA_0200_C2530)

Zusammen mit der Funktion [Length](#) kann auch eine Zeichenfolge mit einer bestimmten Anzahl von Zeichen ab dem ersten Zeichen rechts von einer Zeichenfolge zurückgegeben werden.

Zum Beispiel: die letzten 5 Zeichen einer Zeichenfolge:

```
{[BIPINSTANCE:Typ].Substring([BIPINSTANCE:Typ].Length-5)}
```

Ergibt C2530 (Typname: 173_WT_A_BEWA_0200_C2530)

ToLower()

Umwandlung in Kleinbuchstaben

Ergebnis: Neuer Text

Aus BIMM wird bimm

ToString(Formatierung)

Konvertiert einen beliebigen Wert in eine Zeichenfolge.

Ergebnis: Zeichenfolge

Beispiele:

Zahl (0,2000 m): {[TYPE:Breite].ToString("F2")} ergibt "0,20" als Text

Zahl (0,2000 m): {[TYPE:Breite]*1000}.ToString("0000")} ergibt "0200" als Text

Zahl Währung (27000,00) : {[TYPE:Kosten].ToString("C")} ergibt "27.000,00 €" als Text

Zahl Währung (27000,00) : {[TYPE:Kosten].ToString("EUR #,###0.00")} ergibt "EUR 27.000,00" als Text

Zahl Prozent (0,272): {[TYPE:Faktor].ToString("P1")} ergibt "27,2 %" als Text

ToUpper()

Umwandlung in Großbuchstaben

Ergebnis: Neuer Text
Aus bimm wird BIMM

Konstanten

Momentan gibt es nur eine Konstante: IGNORE

Mit IGNORE können eindeutige Bedingungen in Revit Parameter geschrieben werden, es finden keine Überschreibungen statt (IGNORE macht nur Sinn in IF Anweisungen)

Syntax:

[CONSTANT:Ignore]

Beispiel:

```
{if([BIPINSTANCE:Typ].contains("_BEWA_"),[INSTANCE:Fläche]*2.1,[CONSTANT:Ignore])}
```

```
{if([BIPINSTANCE:Typ].contains("_FASS_"),[INSTANCE:Fläche],[CONSTANT:Ignore])}
```

Die Contains-Methode gibt Wahr oder Falsch zurück. Wenn es also wahr ist, dass im obigen Beispiel im Typnamen die Zeichenfolge "_BEWA_" vorkommt, soll der Wert der Fläche mit 2,1 multipliziert werden. Ansonsten wird der Parameter nicht "angefasst", und kann somit auch nicht überschrieben werden. Das Gleiche gilt für das zweite Beispiel: auch hier wird nur die Wahr-Verzweigung berücksichtigt.

Datum- und Zeitangaben

(Ab Version 2014.2.0.0)

Now

Ruft das aktuelle Datum und die aktuelle Zeit auf dem lokalen Rechner ab

Syntax (Groß- / Kleinschreibung beachten!):

{Now}

Ergebnis: Text

Beispiele (aktuelles Datum: Montag, 18.11.2013):

{Now} ergibt das aktuelle Datum mit Zeit als Text, z.B. "18.11.2013 07:27:22"

{Now.ToString("d")} ... ergibt: 18.11.2013

{Now.ToString("D")} ... ergibt: Montag, 18. November 2013

```
{Now.ToString("dd")} ... ergibt: 18
{Now.ToString("dddd")} ... ergibt: Montag
{Now.ToString("MM")} ... ergibt: 11
{Now.ToString("MMMM")} ... ergibt: November
{Now.ToString("yy")} ... ergibt: 13
{Now.ToString("yyyy")} ... ergibt: 2013
{Now.ToString("HH")} ... ergibt: 07
{Now.ToString("mm")} ... ergibt: 27
{Now.ToString("ss")} ... ergibt: 22
```

Beispiele kombiniert und zusammen mit Text

```
{Now.ToString("yyyyMMdd")} ... ergibt: 20131118
{Now.ToString("yyyy-MM-dd")} ... ergibt: 2013-11-18
{Now.ToString("MMMM'sonne'")} ... ergibt: Novembersonne
(VORSICHT: Text immer innerhalb Apostroph: 'sonne')
```

Konvertierung

(Ab Version 2014.2.0.0)

Nachfolgend sind die wichtigsten Konvertierungsfunktionen abgebildet, die unsere Bibliothek momentan unterstützt.

ToDateTime(Zeichenfolge)

Konvertiert die angegebene Zeichenfolgendarstellung eines Datums und einer Uhrzeit in den entsprechenden Datums- und Uhrzeitwert.

Ergebnis: Datum

ToDouble(Zeichenfolge)

Konvertiert die angegebene Zeichenfolgendarstellung einer Zahl in eine entsprechende Gleitkommazahl .

Ergebnis: Gleitkommazahl

Beispiel:

```
{ToDouble([INSTANCE:Kommentare].SubString(0,3))}
Kommentare: "2,70_RDOK": ergibt 2,7 (Startwert 0)
```

ToInt32(Zeichenfolge)

Konvertiert die angegebene Zeichenfolgendarstellung einer Zahl in eine entsprechende Ganzzahl.

Ergebnis: Ganzzahl

Beispiel:

```
{ToInt32([INSTANCE:Kommentare].SubString(5,2))}
Kommentare: "3,70_00_EG_RDUK": ergibt 0 (Startwert 0)
```

ToString(Zahl)

Konvertiert den Wert einer Zahl in die entsprechende Zeichenfolgendarstellung.

Hinweis: Zahlen werden automatisch in eine Zeichenfolge umgewandelt, wenn der Typ des Zielparameters Text ist.

Ergebnis: Zeichenfolge

Beispiel:

```
{ToString([BIPINSTANCE:Versatz oben])}
```

[BIPINSTANCE:Versatz oben]: 0,2 : ergibt "0.2"

Bedingungen

Bedingungen werden mit **IF** eingeleitet. Es werden drei Argumente benötigt:

Bedingung und jeweilige Verzweigungen falls Bedingung wahr bzw. falsch ist. Die Argumente werden mit einem Komma getrennt (**Vorsicht: Dezimaltrennzeichen ist im PP der Punkt!**)

Die Wahr/Falsch Verzweigungen können mit (momentan) **maximal 7** weiteren Bedingungen verschachtelt werden.

Syntax

```
{if(Bedingung,FallsWahr, FallsFalsch)}
```

Beispiele:

Ja/Nein:

```
{if([INSTANCE:Tragwerk],"WT","WN")}
```

```
{if([INSTANCE:Tragwerk]=true,"WT","WN")}
```

(bzw. =false wenn nein)

```
{if([INSTANCE:Tragwerk]=false,"WN","WT")}
```

oder mit not

```
{if(not[INSTANCE:Tragwerk],"WN","WT")}
```

Verschachtelt:

```
{[INSTANCE:Kommentare].insert(2,[BIPINSTANCE:Typ].substring(4,2))}
```

Verknüpft: (mit AND oder OR)

```
{if([BIPINSTANCE:Typ].startswith("173") OR  
[BIPINSTANCE:Typ].endswith("Wand"),"Ja","Nein")}
```

Vergleiche mit Einheiten: (im PP nur Zahl, ohne Einheit! Standard ist die jeweilige in Revit eingestellte Einheit):

```
{if([INSTANCE:173_010_013_Höhe]>3.2,"Teurer","Günstiger")}
```

Textfunktionen:

```
{if([BIPINSTANCE:Typ].substring(0,4)="173_","Wand","keine Wand")}  
{if([BIPTYPE:Typname].contains("_BEWA_"),"07_02",[CONSTANT:Ignore])}  
{if([INSTANCE:000_080_020_Abschnitt].contains("KW08"),"18.01.2013",[CONSTANT:Ignore])}
```

Verneint: (mit NOT)

```
{if(not[BIPINSTANCE:Typ].substring(0,4)="173_","keine Wand","Wand")}
```

Trigonometrische Funktionen (->[Link zu Wikipedia](#))

Folgende Winkelfunktionen werden unterstützt:

Sin, Cos, Tan, ASin, ACos, ATan, Sinh, Cosh, Tanh

Hinweis: Winkel muss im Bogenmaß angegeben werden; Gradwinkel können jedoch mit $\text{PI}/180$ ins Bogenmaß umgerechnet werden

Beispiel: $\sin 90^\circ (=1)$

```
{sin(90*PI/180)}
```

Sonstiges

Setzen von Ja/Nein Parametern in Revit

Ja/Nein Parameter in Revit werden mit Hilfe von "true" oder "false" innerhalb von Bedingungen gesetzt.

Beispiel:

```
{if([BIPINSTANCE:Typ].contains("173"),true,false)}
```