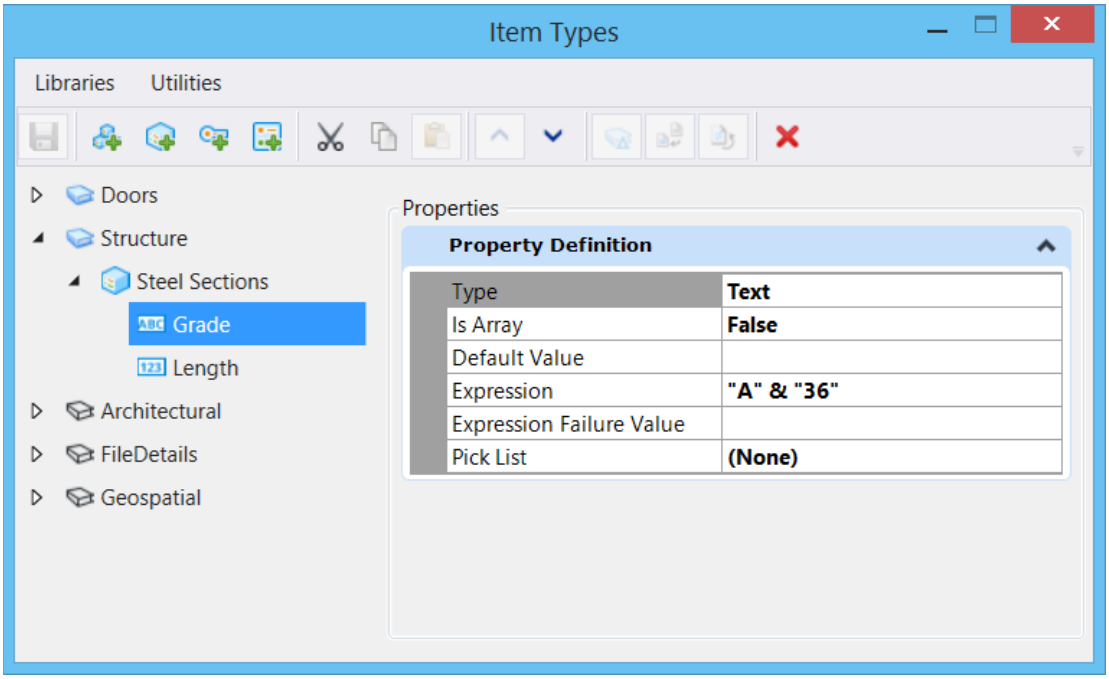
# [Technology Preview] Expression

An expression is a text string that defines the syntax to be evaluated by the expression evaluator. The expression input can comprise of numbers, strings, access strings, symbols, and operators. Once you attach an item type with an expression defined, the corresponding value of the expression will display in the element's Properties dialog.

You can access this feature from the Property Definition section in the Item Types Dialog by selecting an Item's property definition.

Ribbon: Drawing > Attach > Item Types > Item Types dialog launcher.




## Expression components

An expression is built up from left to right, using parentheses to explicitly call functions or group operations. Following are the components of an expression:

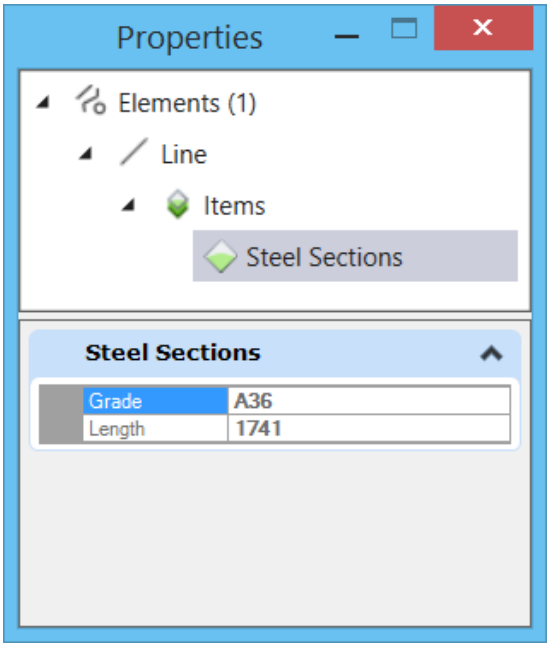
- String - "Pipe"
- Integer - 1
- Double - 6.84
- Symbol - you can use symbols to supply values to expression evaluators. Symbols can be from an instance via an access string, defined value, or an application defined method. Following are a few examples:
    - Instance - "ActiveFile.Author". An access string is a limited expression that contains no blank spaces, operators or variable portions.
    - Value - "System.Math.PI"

    - Method - "System.Math.Sin(1.57)"
- Operators -
    - Comparison - <, >, <=, >=, =, and <>
    - Conditional IIf (conditional, true-result, false-result)
    - Arithmetic ^ (exponentiation), *, /, \, Mod, +, and -. The division operator "/" always produces a result of type double. The division operator "\" always produces an integer result.

# To Define and Apply an Expression

1. Open the Item Types dialog (Drawing > Attach > Item Types dialog launcher).
2. Select the property definition of the desired item.
3. In the Expression field in the Property Definition section, type in the syntax for the required expression.
4. Press `<Enter>` and click Save.
5. Now attach this item with the expression to the desired element.

The expression value will be displayed in the Properties dialog. Refer image below:




**The values for Grade and Length are derived from expressions**

# Expression Types and Syntax Examples

Following are different types of Expressions and their syntax examples:

- General Symbols (Arithmetic)
    - 1 +"4" will return the value 5

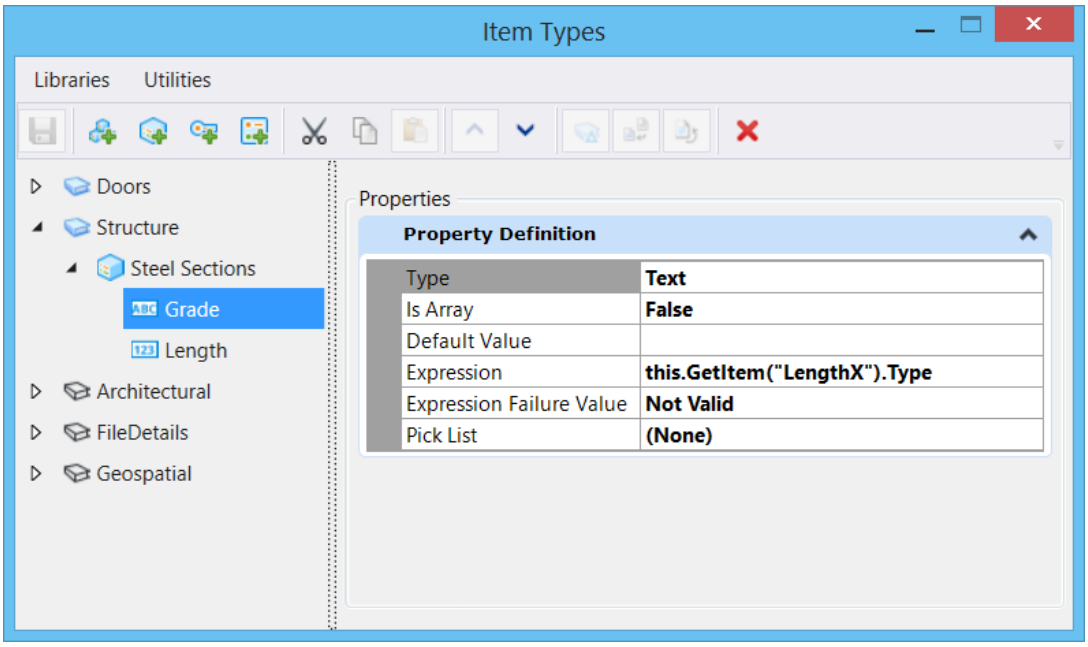
    - 2.3 * 3 will return the value 6.9
    - 12/5 will return the value 2.4
    - 12\5 will return the value 2
    - 25 mod 3 will return the value 1
- String Concatenation
    - 1 & "4" will return the value "14"
    - "Pipe" & "and" & "Valve" will return the value "Pipe and Valve"
- IIf Symbols (Condition)
    - IIf (500>200, "math ok", "math wrong") will return the value "math ok"
    - IIf (500<200, "math ok", "math wrong") will return the value "math wrong"
- IIf Smbol (Comparison)
    - 20<10 will return the value "false"
    - IIf(System.String.Compare ("Pipe", "Pipe"), "match", "no match") will return the value "match"
- SystemString
    - System.String.Length("Pipe") will return the value "4"
- System Math
    - System.Math.Sin(45*System.Math.PI/180)
    - System.Math.Round(4.53459)
    - System.Math.Log(1.2) / System.Math.Log(0.1)
    - System.Math.Max(1.2, 1.1)
- Element Properties
    - this.GetElement().ElementID
    - this.GetElement().TotalLength
    - this.GetElement().ElementDescription
    - this.GetElement().Color
- Model Properties
    - this.GetModel().Is3D
    - this.GetModel().Name
    - this.GetModel().Description
    - this.GetModel().Resolution
- File Properties
    - this.GetFile().Author
    - this.GetFile().Editor
    - this.GetFile().FileName
- GetItemSameLibrary()
    - this.GetItem("Lib1Itp1").TextProp
    - this.GetItem("Lib1Itp1").IntegerProp
    - this.GetItem("Lib1Itp1").DoubleProp
    - this.GetItem("Lib1Itp1").ArrayProp[0] & " \" & this.GetItem("Lib1Itp1").ArrayProp[1]
- GetItemDifferentLibrary()
    - this.GetItem("Lib1:Lib1Itp1").TextProp
    - this.GetItem("Lib1:Lib1Itp1").IntegerProp
    - this.GetItem("Lib1:Lib1Itp1").ArrayProp[0] & " \" & this.GetItem("Lib1:Lib1Itp1").ArrayProp[1]

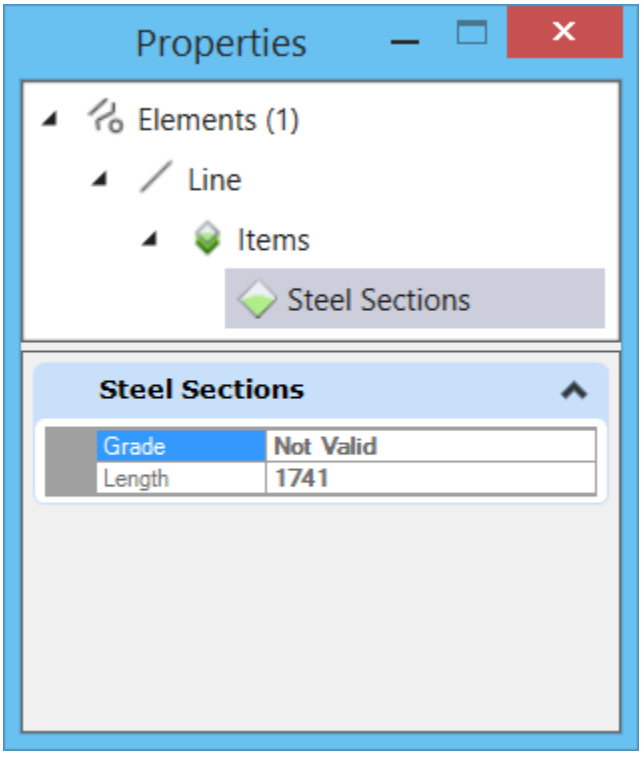
- GetItemExtrinsicSchema
  - this.GetItem("DGNECPlugin_Test:Father").FirstName
  - this.GetItem("DGNECPlugin_Test:Father").FirstName & " \" & this.GetItem("DGNECPlugin_Test:Father").LastName
- GetItemRelatedExtrinsicSchema
  - this.GetItem("DGNECPlugin_Test:Father").FirstName & " \" & this.GetItem("DGNECPlugin_Test:Father").LastName"
  - this.GetItem("DGNECPlugin_Test:Father").GetRelatedItem("FatherHasChild:0:Child").Tag

- GetDisplayString
  - this.GetElement().GetDisplayString(\"TotalLength")
  - this.GetElement().GetDisplayString(\"ElementID\")
  - this.GetElement().GetDisplayString(\"Segments[0].Length\")
  - this.GetElement().GetDisplayString(\"Segments[0].Direction\")
  - this.GetElement().GetDisplayString(\"Level\")
  - this.GetElement().GetDisplayString(\"Color\")
  - this.GetElement().GetDisplayString(\"Style\")
  - this.GetElement().GetDisplayString(\"DisplayStyle\")
  - this.GetModel().GetDisplayString(\"MasterUnit\")
  - this.GetModel().GetDisplayString(\"SubUnit\")
  - this.GetModel().GetDisplayString(\"AngleReadoutFormat\")
- GetDisplayValue
  - this.GetElement().GetDisplayValue(\"ElementID\")
  - this.GetElement().GetDisplayValue(\"TotalLength\")
  - this.GetElement().GetDisplayValue(\"Segments[0].Length\")
  - this.GetElement().GetDisplayValue(\"Segments[0].Direction\")
  - this.GetElement().GetDisplayValue(\"Level\")
  - this.GetElement().GetDisplayValue(\"Color\")
  - this.GetElement().GetDisplayValue(\"Style\")
  - this.GetElement().GetDisplayValue(\"DisplayStyle\")
  - this.GetModel().GetDisplayValue(\"MasterUnit\")
  - this.GetModel().GetDisplayValue(\"SubUnit\")
  - this.GetModel().GetDisplayValue(\"AngleReadoutFormat\")

# Expression Failure Value

In this field you can type in the value that will be displayed in the item's properties if the expression fails to execute. You can type in the desired failure value in the Expression Failure Value field in the Properties Definition section of Item Types dialog.




**Grade - displays the failure value "Not Valid" since "LengthX" is not a valid entry in the expression**