TABLE OF CONTENTS

1. CODE CHECKING IN BIM

Code Checking is a crucial stage of Structural Engineering Design Process. It is intended to facilitate the Code Checking workflows from Revit to take advantage and contribute to accurate definition of Design Models and also to deliver prerequisites for Structural Detailing stages to be subsequently performed in this authoring enviroment. This is with enriching Revit models with appropriate information and delivering appropriate data and documentation in sequence.

In order to drive the Code Checking process in Revit environment, there are following enablers that need to be present:

- Precise Physical Geometry
- Structural Analytical Results (Internal Forces, Stresses, Deflections, etc.)
- User Interface to drive general Code Checking tasks in Revit (Code Checking Framework)
- Specific Code implementation

Precise Physical geometry is naturally represented by Revit Design model and structural elements definition.

Structural Analytical Results can be stored in Revit models by Structural Analysis products (see also *Autodesk® Robot™ Structural Analysis Professional*) or services (See *Structural Analysis for Autodesk® Revit*)

User Interface to drive general Code Checking tasks is delivered with installation of *"Structural Analysis and Code checking Toolkit for Autodesk® Revit"*. This is a framework that allows specific codes to be activated and being operated by end users to service expected workflows.

Specific Code implementation is assumed to be a separate deliverable, an add-in for Revit, that will assure factual Code Checking capability in accordance to selected country code regulations. The Code implementation shall contain code parameters definition, computational algorithms for design and verification of structural elements as well as reporting capabilities. The results of the Code Checking add-ins may be delivered in a form of Structural Results (visualized in a form of diagrams and maps), calculation notes, model and documentation updates (elements sizing, 3D Reinforcement, Schedules, Fabrication documentation, etc..)

The purpose of this documentation is to help in development of the latter component.

2. WHAT IS CODE CHECKING SDK?

2.1. What is Code Checking Framework for Revit users?

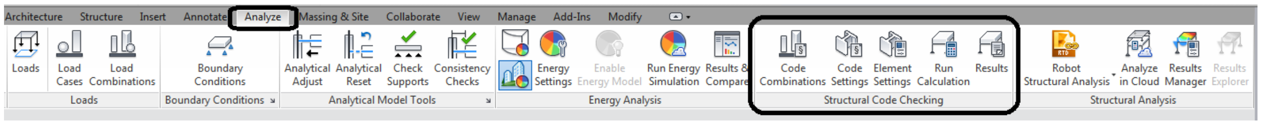
The Structural Code Checking Framework is a new Revit feature to give Revit users the ability to perform a code checking process inside a native Revit model.

Revit users will now be able to define some rules for an overall Code checking application and a set of parameters defining expected behavior for individual analytical model elements or group of analytical model elements. These parameters will extend Revit model and can be used as attributes for the calculation process itself.

- **Ribbon:**

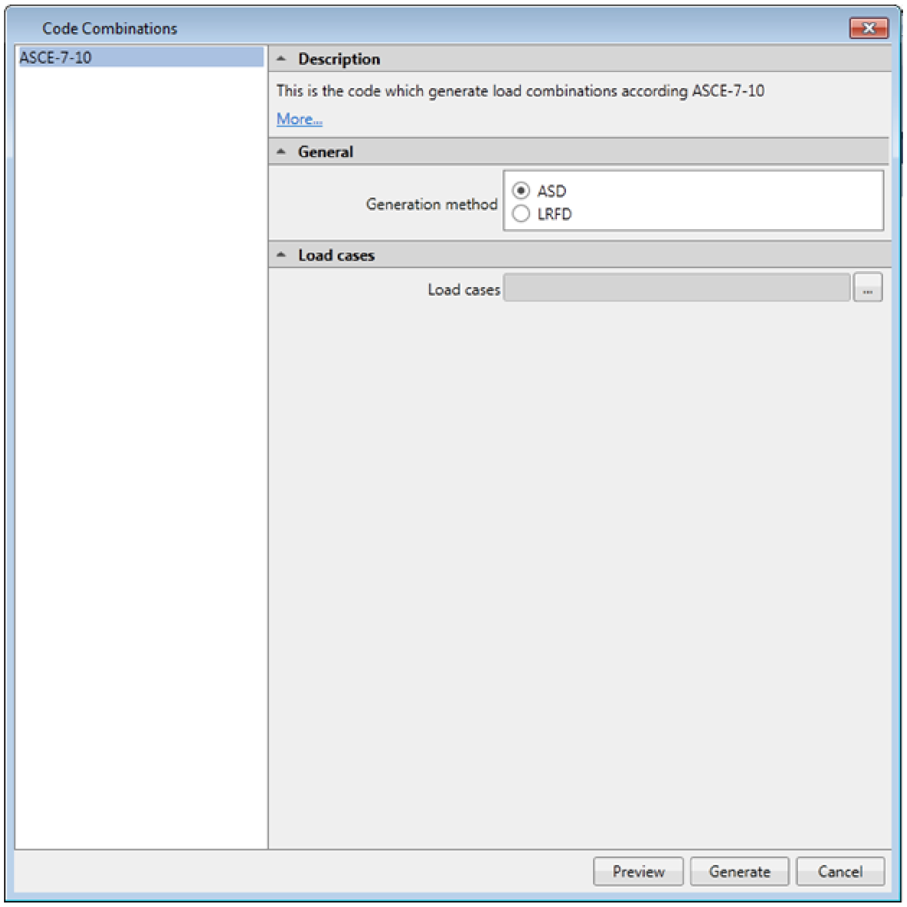
For that purpose, a set of new commands dedicated to Structural Code Checking have been exposed.

These commands could be found on the ribbon tab Analyze, on the panel Structural Code Checking when as minima one code is installed.



- **Command Code Combinations:**

The Code Combination dialog is a placeholder to activate code combinations to give Revit users the ability to generate load combinations based on specific regulations.



This is the code which generate load combinations according ASCE-7-10

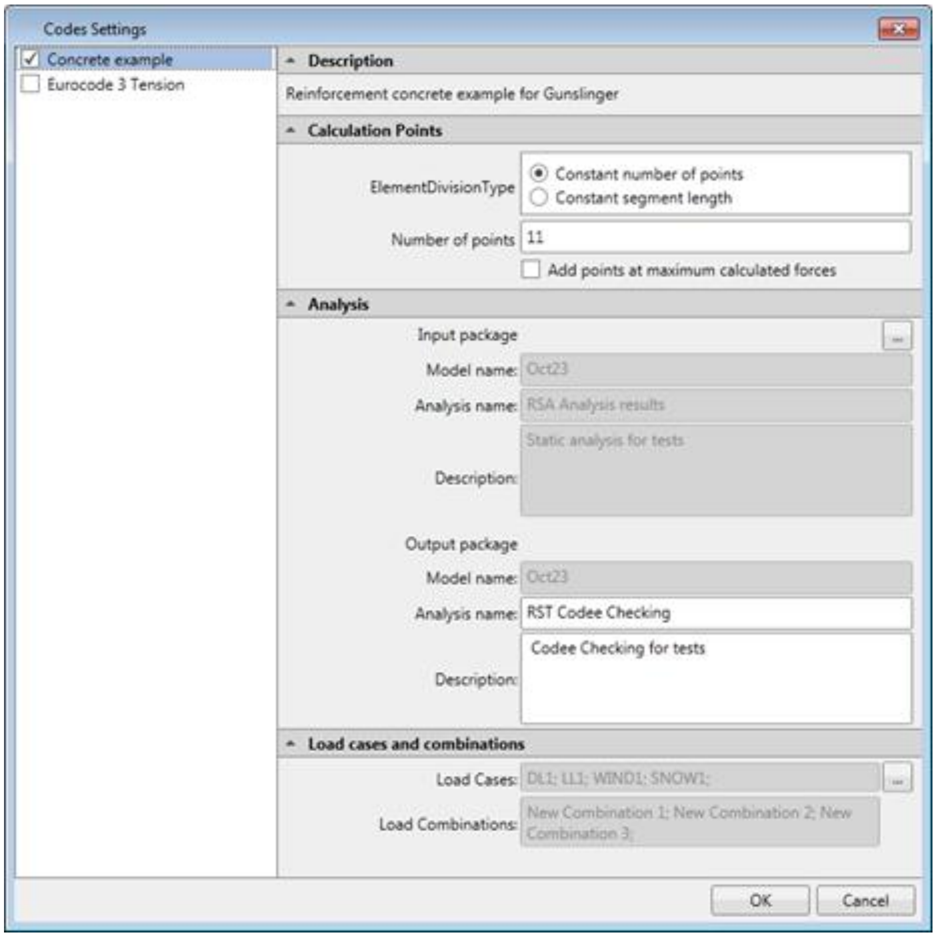
Generation method: [✓ ASD] LRFD

Load cases

- **Command Code Settings:**

The Code Settings dialog is a placeholder to activate codes available and to define some global settings, common for all elements which will be calculated. These settings could be:

- Code specifics parameters and calculations options to take into consideration
- Structural analysis package to use as input data
- Load cases and combinations available in Revit model that will be used for the calculation process


Codes Settings

[✓] Concrete example
[ ] Eurocode 3 Tension

Description

Reinforcement concrete example for Gunslinger

Calculation Points

ElementDivisionType: [● Constant number of points] ○ Constant segment length

Number of points: 11

[ ] Add points at maximum calculated forces

Analysis

Input package

Model name: Oct23

Analysis name: RSA Analysis results

Description: Static analysis for tests

Output package

Model name: Oct23

Analysis name: RST Codee Checking

Description: Codee Checking for tests

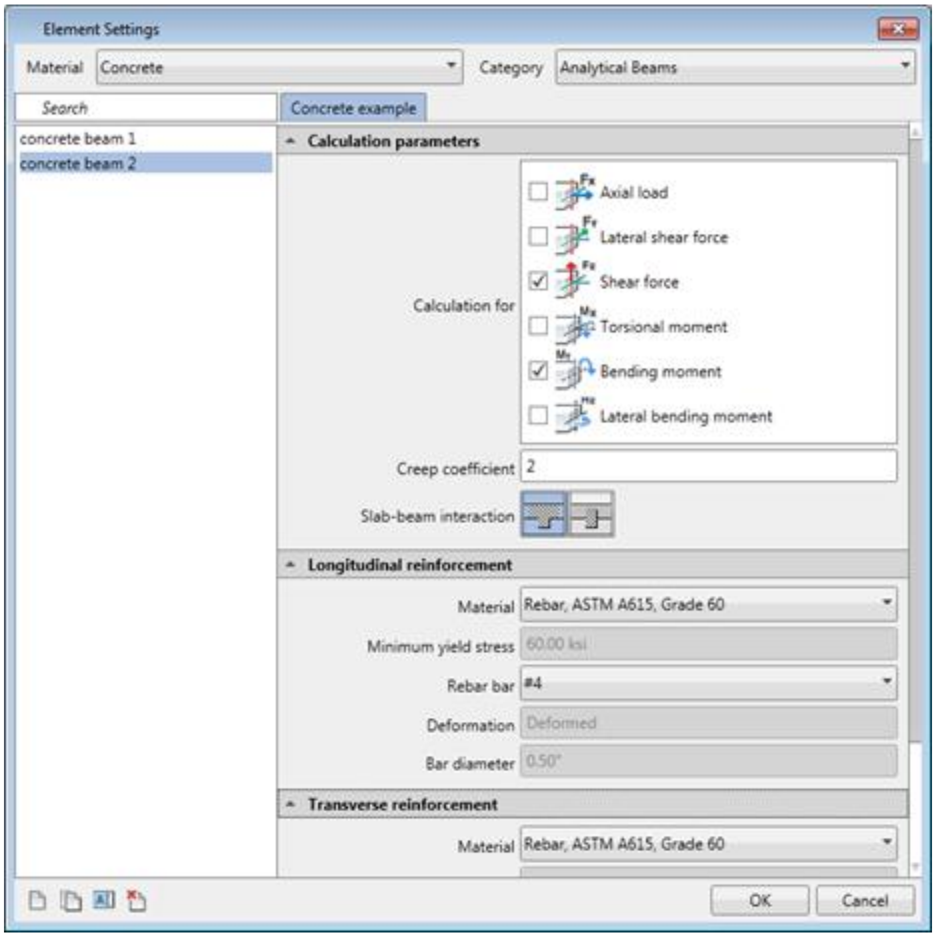
Load cases and combinations

Load Cases: DL1; LL1; WIND1; SNOW1;

Load Combinations: New Combination 1; New Combination 2; New Combination 3;

- **Command Element Settings:**

The Element Settings dialog is the placeholder to define and manage a code specific set of parameters that could be applied to elements before running calculations.



Element Settings

Material: Concrete — Category: Analytical Beams

Search

- concrete beam 1
- concrete beam 2

Concrete example

Calculation parameters

Calculation for:
- [ ] Axial load
- [ ] Lateral shear force
- [x] Shear force
- [ ] Torsional moment
- [x] Bending moment
- [ ] Lateral bending moment

Creep coefficient: 2

Slab-beam interaction

Longitudinal reinforcement

Material: Rebar, ASTM A615, Grade 60

Minimum yield stress: 60.00 ksi

Rebar bar: #4

Deformation: Deformed

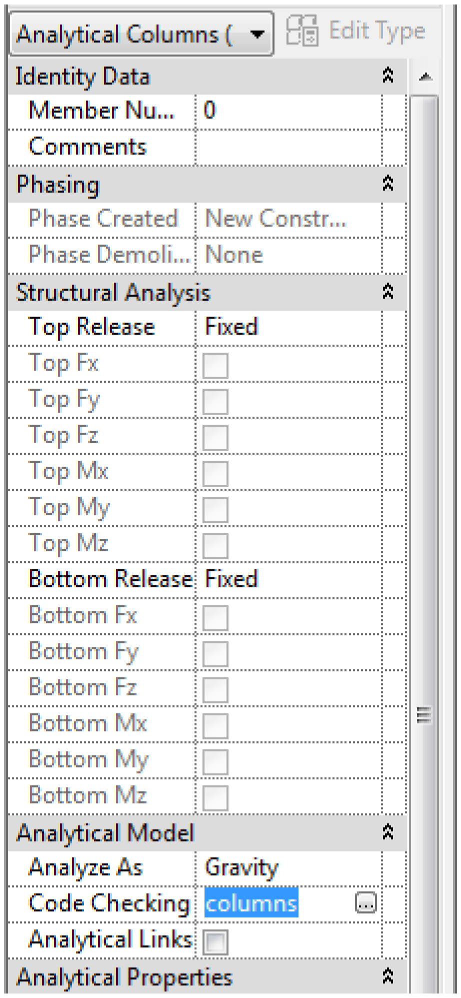
Bar diameter: 0.50"

Transverse reinforcement

Material: Rebar, ASTM A615, Grade 60

OK — Cancel

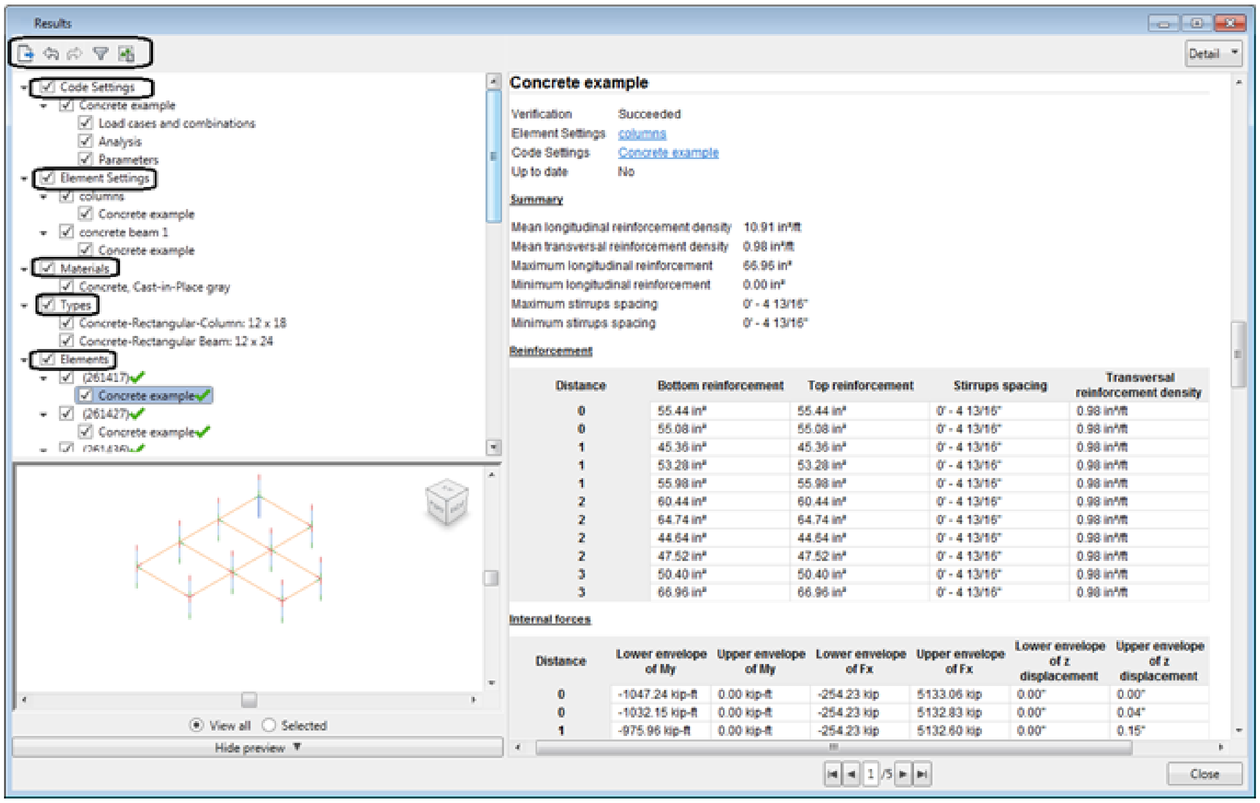
To assign a set of parameter to Revit element(s), this dialog could be called via the Code Checking Parameter available for all analytical model element categories on the Properties Palette.



Identity Data	
Member Nu...	0
Comments	
Phasing	
Phase Created	New Constr...
Phase Demoli...	None
Structural Analysis	
Top Release	Fixed
Top Fx	☐
Top Fy	☐
Top Fz	☐
Top Mx	☐
Top My	☐
Top Mz	☐
Bottom Release	Fixed
Bottom Fx	☐
Bottom Fy	☐
Bottom Fz	☐
Bottom Mx	☐
Bottom My	☐
Bottom Mz	☐
Analytical Model	
Analyze As	Gravity
Code Checking	columns
Analytical Links	☐
Analytical Properties	

- **Command Run Calculations:**

The Run Calculations command will perform calculations based on selected code(s) and family instance(s).
At the end of calculation, results are exposed on a report dialog.



- Code Settings
  - Concrete example
    - Load cases and combinations
    - Analysis
    - Parameters
- Element Settings
  - columns
    - Concrete example
  - concrete beam 1
    - Concrete example
- Materials
  - Concrete, Cast-in-Place gray
- Types
  - Concrete-Rectangular-Column: 12 x 18
  - Concrete-Rectangular Beam: 12 x 24
- Elements
  - (261417)✓
    - Concrete example✓
  - (261427)✓
    - Concrete example✓

Concrete example

Verification	Succeeded
Element Settings	columns
Code Settings	Concrete example
Up to date	No

Summary

Mean longitudinal reinforcement density	10.91 in^2/ft
Mean transversal reinforcement density	0.98 in^2/ft
Maximum longitudinal reinforcement	66.96 in^2
Minimum longitudinal reinforcement	0.00 in^2
Maximum stirrups spacing	0' - 4 13/16"
Minimum stirrups spacing	0' - 4 13/16"

Reinforcement

Distance	Bottom reinforcement	Top reinforcement	Stirrups spacing	Transversal reinforcement density
0	55.44 in^2	55.44 in^2	0' - 4 13/16"	0.98 in^2/ft
0	55.08 in^2	55.08 in^2	0' - 4 13/16"	0.98 in^2/ft
1	45.36 in^2	45.36 in^2	0' - 4 13/16"	0.98 in^2/ft
1	53.28 in^2	53.28 in^2	0' - 4 13/16"	0.98 in^2/ft
1	55.98 in^2	55.98 in^2	0' - 4 13/16"	0.98 in^2/ft
2	60.44 in^2	60.44 in^2	0' - 4 13/16"	0.98 in^2/ft
2	64.74 in^2	64.74 in^2	0' - 4 13/16"	0.98 in^2/ft
2	44.64 in^2	44.64 in^2	0' - 4 13/16"	0.98 in^2/ft
2	47.52 in^2	47.52 in^2	0' - 4 13/16"	0.98 in^2/ft
3	50.40 in^2	50.40 in^2	0' - 4 13/16"	0.98 in^2/ft
3	66.96 in^2	66.96 in^2	0' - 4 13/16"	0.98 in^2/ft

Internal forces

Distance	Lower envelope of My	Upper envelope of My	Lower envelope of Fx	Upper envelope of Fx	Lower envelope of z displacement	Upper envelope of z displacement
0	-1047.24 kip-ft	0.00 kip-ft	-254.23 kip	5133.06 kip	0.00"	0.00"
0	-1032.15 kip-ft	0.00 kip-ft	-254.23 kip	5132.83 kip	0.00"	0.04"
1	-975.96 kip-ft	0.00 kip-ft	-254.23 kip	5132.60 kip	0.00"	0.15"

Some additional tools are exposed on the menu of this dialog:

- Export the report as html and Mht
- Filter elements based on their status
- Create a Revit selection that could be reused

Inside the navigation tree could be found the settings used to performed calculations and some information related to Revit model (i.e. family instance and material). All calculated elements are listed with a specific status.

The report and the 3D viewer are synchronized with the navigation tree.

- **Command Results:**

The command Results allow to retrieve calculations results for a specific set of elements.

From Revit Users perspective, to activate this functionality the Structural Analysis and Code checking Toolkit for Autodesk Revit available for free from the Exchange app website and a Code Checking Application should be installed on the client machine.

2.2. What is Code Checking Framework for Developers?

To enable the Code Checking workflow for Revit users, some Code Checking applications should be created. To achieve this, the Code Checking Framework should be leverage by Revit API developers.

From Revit API developer perspective, the Code Checking Framework is an API platform supporting development of Structural Code Checking Applications for Revit and making them consistent and aligned with the way Revit interacts with users.

The primary goal of the Code Checking Framework is to help Revit API developers concentrate on essential development aspects when creating Structural Code Checking Applications for Revit, by providing support for typical, commonly used functionalities.

The outcome of this approach is:

- A full and consistent integration of Structural Code Checking Applications within the Revit environment (UI and behaviors)
- Acceleration of Code Checking Applications development by providing a set of tools and components for typical Structural Code Checking Application development.
- Enabling easily Structural Code Checking Applications activation and registration by Revit users

2.3.What is Code Checking Framework SDK?

The Code Checking SDK is a development environment for Rapid Application Development purposes that helps to create, deploy and activate add-ins based on the Extensible Storage Framework, Code Checking Framework and engineering components, Results builder and Revit APIs.

The core part of the SDK is implemented as a form of Microsoft Visual Studio C# templates. Using templates provided, developers can quickly build and deploy an application that has a similar look & feel and behaviors to other Code Checking solution working on top of Revit.

The Code Checking Framework *SDK takes advantage of the Revit API and of a set of components including UI layout creation feature, advanced controls, Extensible Storage, External Services and structural engineering components.*

The Code Checking Framework *SDK is composed of:*

- Visual Studio tools (C# templates and a Visual Studio addin)
    - Data management and serialization
    - UI definition
    - User interactions
    - Report generation
- Documentation
    - Getting Started
    - User Manual
    - Samples and associated documentation
    - API documentation (chm)

3 types of C# templates are provided with this SDK to create some external commands using the Extensible Storage Framework, some generic Code Checking applications and some Code Checking applications dedicated to required reinforcement for concrete beams and columns.

To support the creation of Visual Studio projects based on templates, the SDK includes a Visual Studio addin. This addin provides also a property generation feature to facilitate data definition.

2.4. To whom is it addressed?

The advantages of this technology can be useful to all Revit API developers making Structural Code Checking Applications for Revit, but it’s most efficient for those whom the following aspects are applicable:

- Create commercial Code Checking Applications for further distribution
- Developing links with other products

2.5. What are the most interesting features for API developers?

- C# project templates ready to build
- Full and consistent integration with Revit
- Automated class storage and visualization within BIM models using Extensible Storage
- Common UI controls and Layout builder
- HTML report generation
- Units display and editing consistently with Revit
- Engineering components to extract from BIM models necessary data to perform Code Checks
- Engineering component for reinforcement design
- Open to integrate previous developed calculation engines.

2.6. What are the benefits for Revit users?

The primary advantages for end-users are:

- Consistent look & feel across various Code Checking Applications
- Consistent behavior aligned with Revit product
- Perform Code Checking in BIM models.

2.7. What is the difference between the Revit and the Code Checking Framework API?

The Code Checking SDK does not provide any additional access to internal Revit functionality.
The Code Checking Applications API extends the Revit API functionality with a set of new components.

2.8. What products can this technology be applied to?

The Code Checking Applications based approach is applicable for Revit Structure and Revit products.

2.9. When are alternative approaches recommended?

The Code Checking SDK provides a set of tools to develop Structural Code Checking Applications on top of the Revit API.

The Code Checking Framework is dedicated to Structural Code Check Applications.
Calculations could be performed on elements having a structural material assigned and of following Categories:

- OST_StructConnnections
- OST_StructuralColumns
- OST_StructuralFoundations
- OST_StructuralFraming
- OST_Floor
- OST_Wall
- OST_StructuralTrusses
- OST_StructuralFramingSystem

Another limitation of the Code Checking SDK lies in the supported language: project templates and examples are provided only in C# and are not applicable for development in other .NET compatible languages however API exposed could be used form any .NET languages.

2.10. How do I get started?

This document describes how to create a first and simple application for Revit. The next stage will be to create a more complex one. For this, the examples and the user manual will be useful to understand features exposed via the Code Checking SDK and how to use them.

3. SYSTEM REQUIREMENTS

- Visual Studio 2012
- .NET Framework 4.5
- Revit 2015
- Structural Analysis and Code checking Toolkit for Autodesk Revit

4. CONFIGURATION

To properly configure Visual Studio 2012, templates should be copied to dedicated folders

- The content of the folder "..\Software Development Kit \Structural Analysis SDK\Visual Studio\Templates\Items\" should be copied to "C:\Users\<current user>\Documents\Visual Studio 2012\Templates\ItemTemplates\Visual C#\Autodesk\Code Checking\"

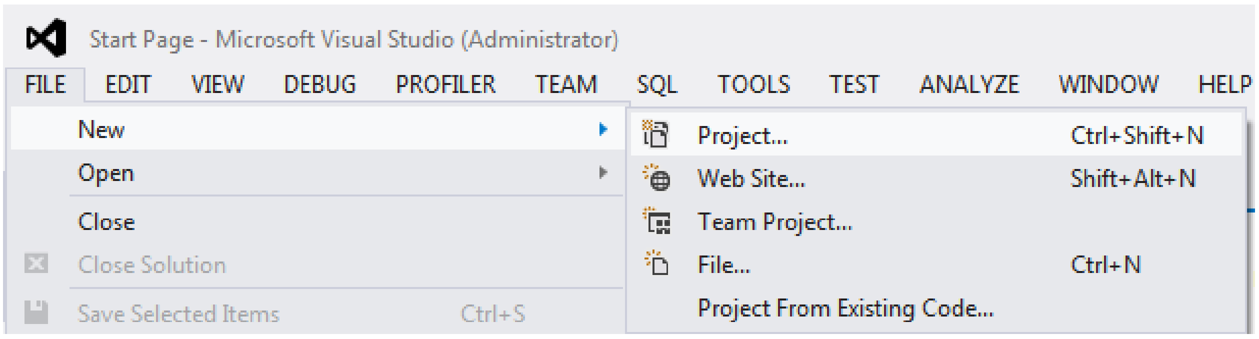
- The content of the folder "..\Software Development Kit \ Structural Analysis SDK\Visual Studio\ Templates\Projects\" should be copied to "C:\Users\<current user>\Documents\Visual Studio 2012\Templates\ProjectTemplates\Visual C# Autodesk\Code Checking\"

- The content of the folder "..\Software Development Kit\Structural Analysis SDK\Visual Studio\Addins" should be copied to "C:\Users\<current user>\Documents\Visual Studio 2012\Addins\" (if this folder doesn't exist, it should be created)

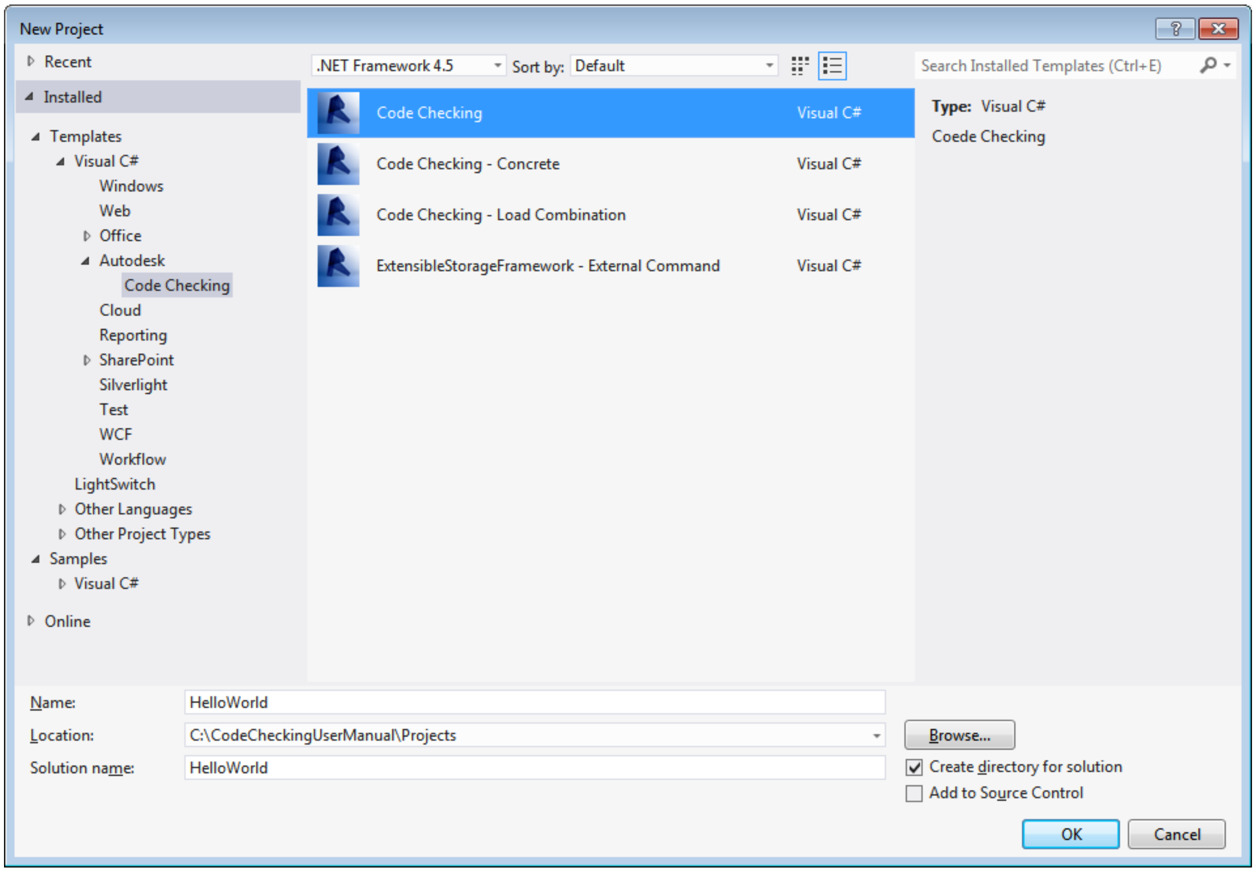
5. FIRST PROJECT CREATION

5.1. Visual Studio solution

- After starting visual studio, you need to choose Autodesk\ Code Checking Templates as a new project (**File / New / Project**).

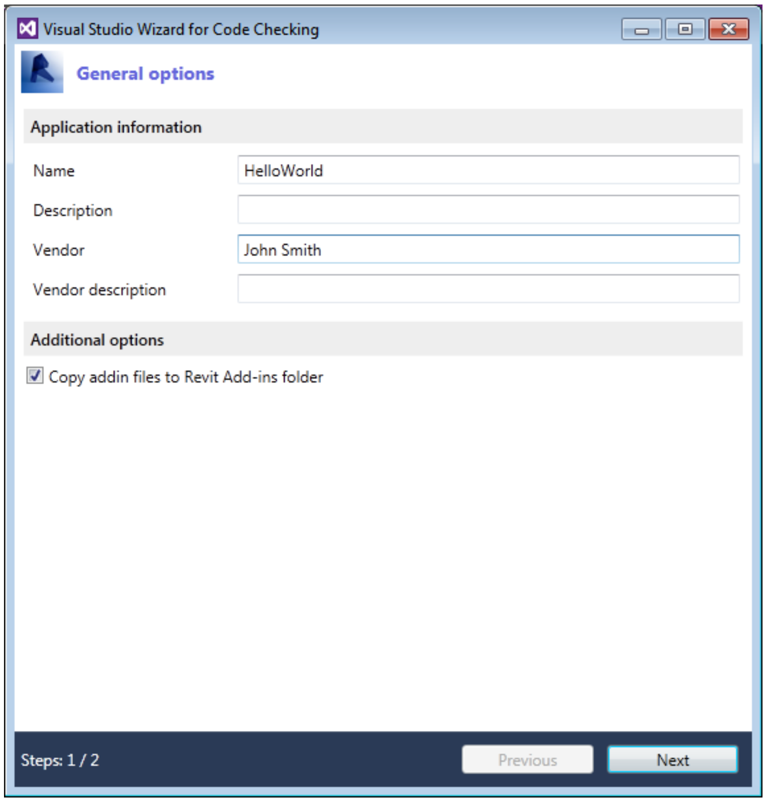


- The dialog below will appear.



Name: HelloWorld
Location: C:\CodeCheckingUserManual\Projects
Solution name: HelloWorld
[✓] Create directory for solution
[ ] Add to Source Control

- At this stage, developer needs to specify which type of Code Checking template he would like to use, a new project name and the folder where he wants to save his source code.



Application information

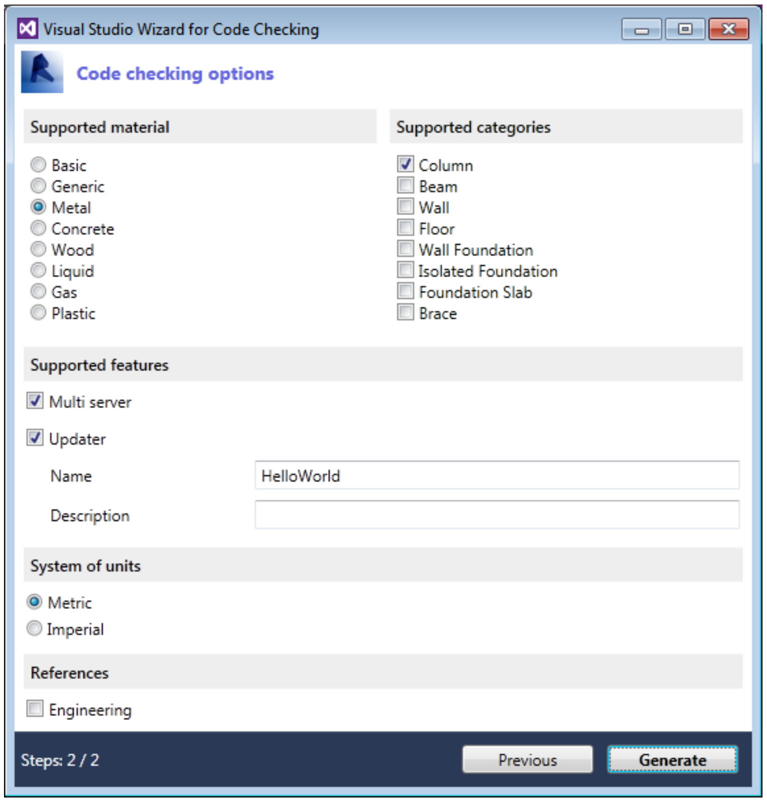
Name	HelloWorld
Description	
Vendor	John Smith
Vendor description	

Additional options

- [✓] Copy addin files to Revit Add-ins folder

Steps: 1 / 2

- After validation, a set of options could be defined:



Supported material

- ○ Basic
- ○ Generic
- ● Metal
- ○ Concrete
- ○ Wood
- ○ Liquid
- ○ Gas
- ○ Plastic

Supported categories

- [✓] Column
- [ ] Beam
- [ ] Wall
- [ ] Floor
- [ ] Wall Foundation
- [ ] Isolated Foundation
- [ ] Foundation Slab
- [ ] Brace

Supported features

- [✓] Multi server
- [✓] Updater

Name	HelloWorld
Description	

System of units

- ● Metric
- ○ Imperial

References

- [ ] Engineering

Steps: 2 / 2

- In this case, our new application will support analytical column and steel material.
- The Visual Studio solution will be created after clicking on the Generate button with all needed references and source files. Manifest files for the new application will be copied to the regular Revit addin user folder.

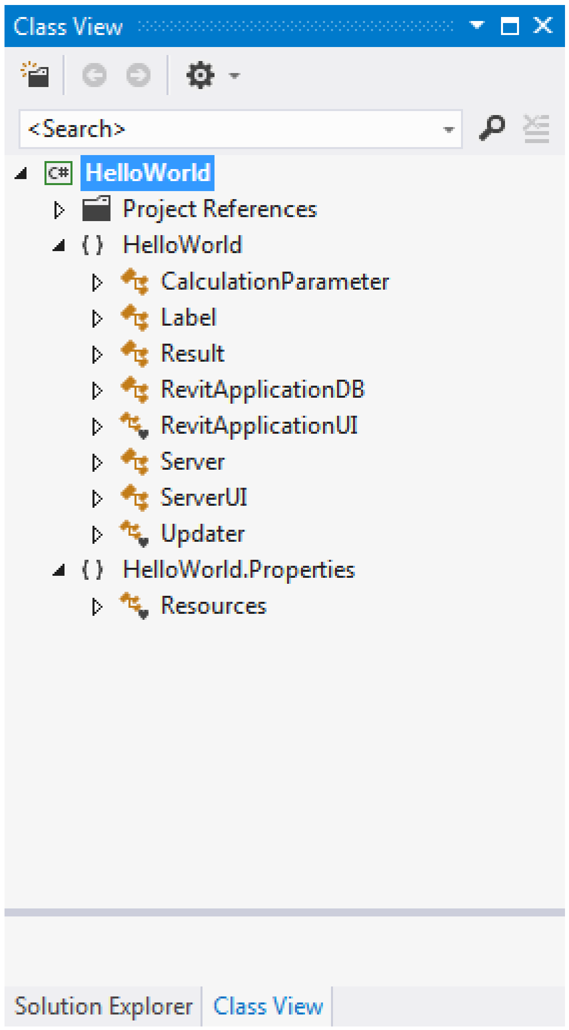
5.2. Development

Goal of this example will be to expose a parameter “A” on the “Code Settings” dialog, a parameter “B” on the “Element Settings” dialog that could be applied to steel column and as result expose the sum of A and B inside the calculation report after calculations.

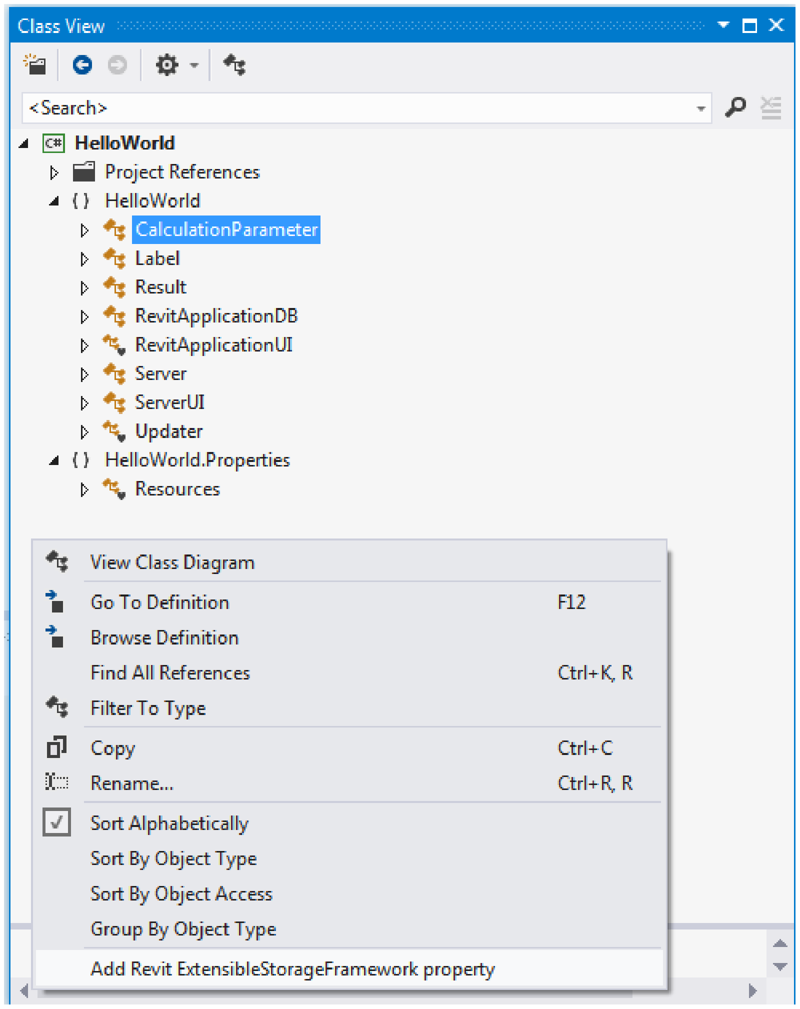
5.2.1. A parameter

To properly define “A” parameter:

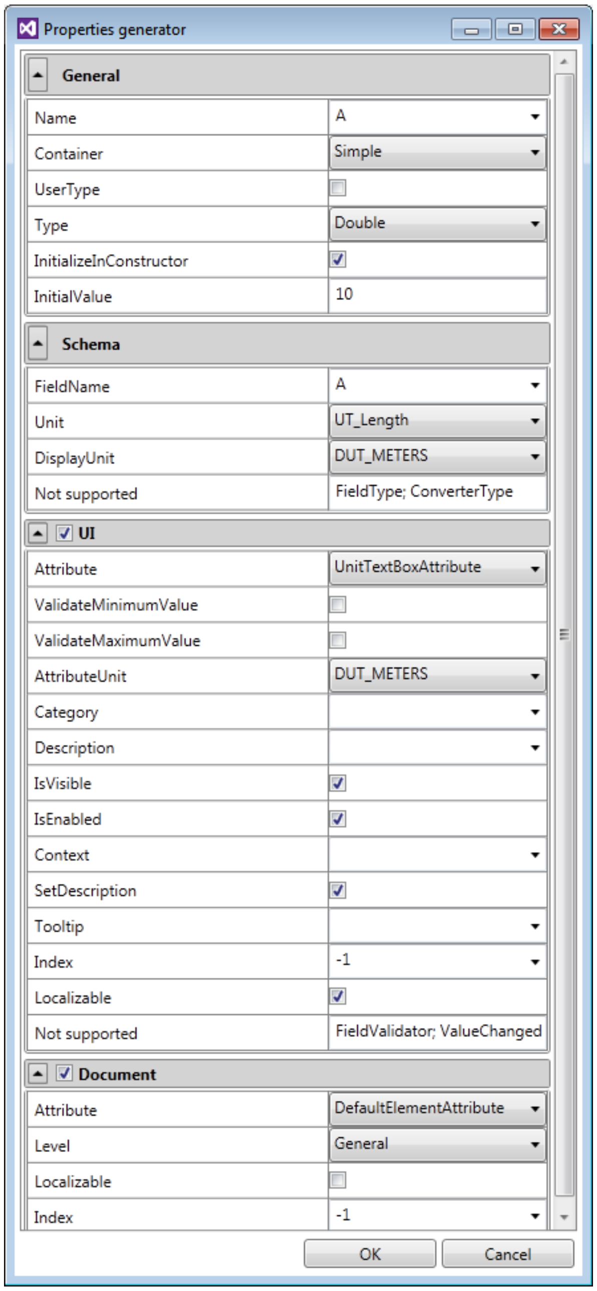
- Go to class view and right click on CalculationParameter



- Select the VS addin to create a schema class



- Define “A” parameter by setting properties as follow and shown on the next screen capture:
  - Name: A
  - Container: Simple
  - Type: Double
  - InitializeInConstructor: true
  - InitialValue: 10
  - Fieldname: A
  - On UI, select a UnitTextBox as type of attribute



Properties generator

General

Name	A
Container	Simple
UserType	☐
Type	Double
InitializeInConstructor	☑
InitialValue	10

Schema

FieldName	A
Unit	UT_Length
DisplayUnit	DUT_METERS
Not supported	FieldType; ConverterType

☑ **UI**

Attribute	UnitTextBoxAttribute
ValidateMinimumValue	☐
ValidateMaximumValue	☐
AttributeUnit	DUT_METERS
Category	
Description	
IsVisible	☑
IsEnabled	☑
Context	
SetDescription	☑
Tooltip	
Index	-1
Localizable	☑
Not supported	FieldValidator; ValueChanged

☑ **Document**

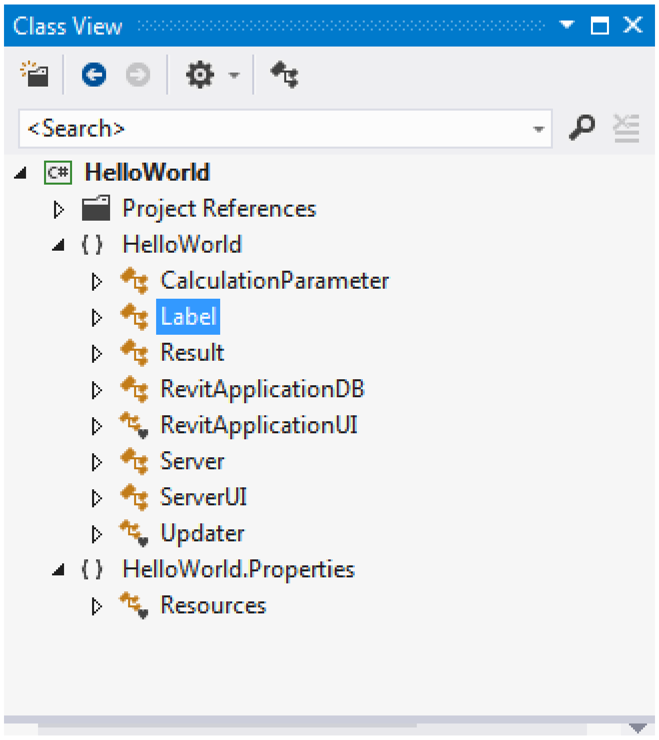
Attribute	DefaultElementAttribute
Level	General
Localizable	☐
Index	-1

- Click OK
- Source code is generated inside CalculationParameters.cs file.

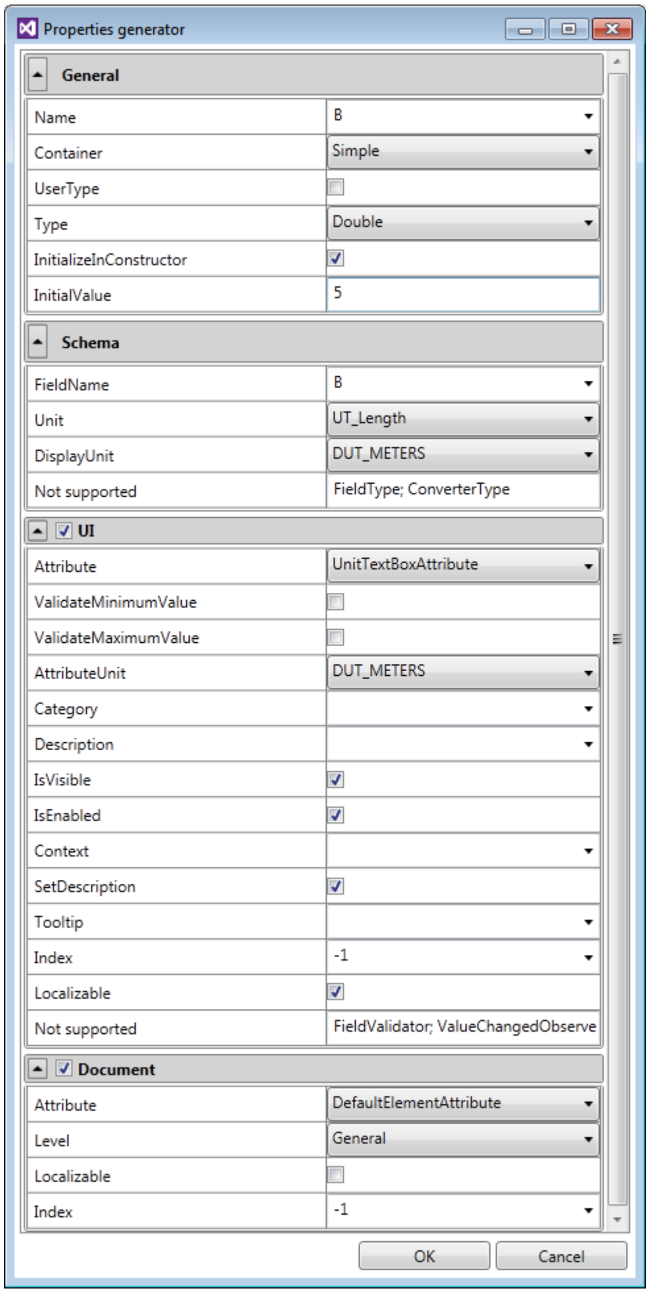
5.2.2. B parameter

To define “B” parameter:

- Select the Label class on the class view



- Define "B" parameter by setting properties as follow and shown on the next screen capture:
  - Name: B
  - Container: Simple
  - Type: Double
  - InitializeInConstructor: true
  - InitialValue: 5
  - Fieldname: B
  - On UI, select a UnitTextBox as type of attribute


Properties generator

General

Name	B
Container	Simple
UserType	☐
Type	Double
InitializeInConstructor	✓
InitialValue	5

Schema

FieldName	B
Unit	UT_Length
DisplayUnit	DUT_METERS
Not supported	FieldType; ConverterType

✓ **UI**

Attribute	UnitTextBoxAttribute
ValidateMinimumValue	☐
ValidateMaximumValue	☐
AttributeUnit	DUT_METERS
Category	
Description	
IsVisible	✓
IsEnabled	✓
Context	
SetDescription	✓
Tooltip	
Index	-1
Localizable	✓
Not supported	FieldValidator; ValueChangedObserve

✓ **Document**

Attribute	DefaultElementAttribute
Level	General
Localizable	☐
Index	-1

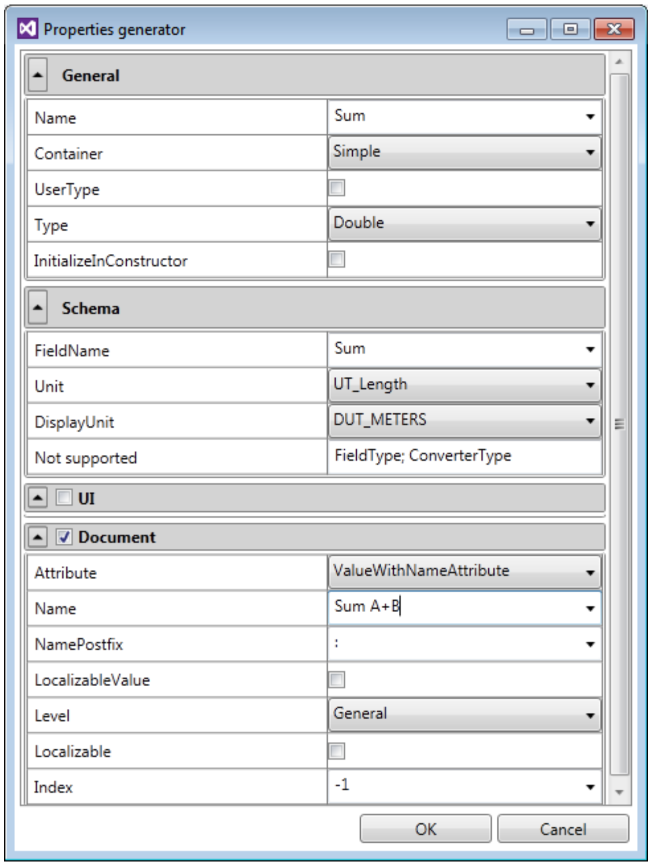
- Click OK
- Source code is generated inside Label.cs file.

5.2.3. Sum Parameter

To define the “Sum” parameter:

- Select the Result class on the class view
- Define “Sum” parameter by setting properties as follow and shown on the next screen capture:
  - Name: Sum
  - Container: Simple
  - Type: Double
  - Fieldname: B

- Disable the UI to expose this parameter only on the report
- On Document, select a ValueWithNameAttribute as type of attribute



Properties generator

General

Name	Sum
Container	Simple
UserType	☐
Type	Double
InitializeInConstructor	☐

Schema

FieldName	Sum
Unit	UT_Length
DisplayUnit	DUT_METERS
Not supported	FieldType; ConverterType

☐ **UI**

☑ **Document**

Attribute	ValueWithNameAttribute
Name	Sum A+B
NamePostfix	:
LocalizableValue	☐
Level	General
Localizable	☐
Index	-1

OK Cancel

- Click OK
- Source code is generated inside Result.cs file.

5.2.4. Calculation

To perform calculation:

- Go to Server.cs file
- Locate the `verify()` method and add the sum operation as follow

Code region

```
public override void Verify(Autodesk.Revit.DB.CodeChecking.ServiceData data)
{
       …
                        Result myResult = new Result();
                        // line to add
                        myResult.Sum = myParams.A + myLabel.B;
       …
}
```

- To simplify this example, structural analysis results and load cases won't be used. To achieve this, two methods have to be changed to return false:

Code region

```
public override bool LoadCasesAndCombinationsSupport()
{
            //line to change
            //return true;
            return false;
}

public override bool ResultBuilderPackagesAsInputData()
{
            //line to change
            //return true;
            return false;
}
```

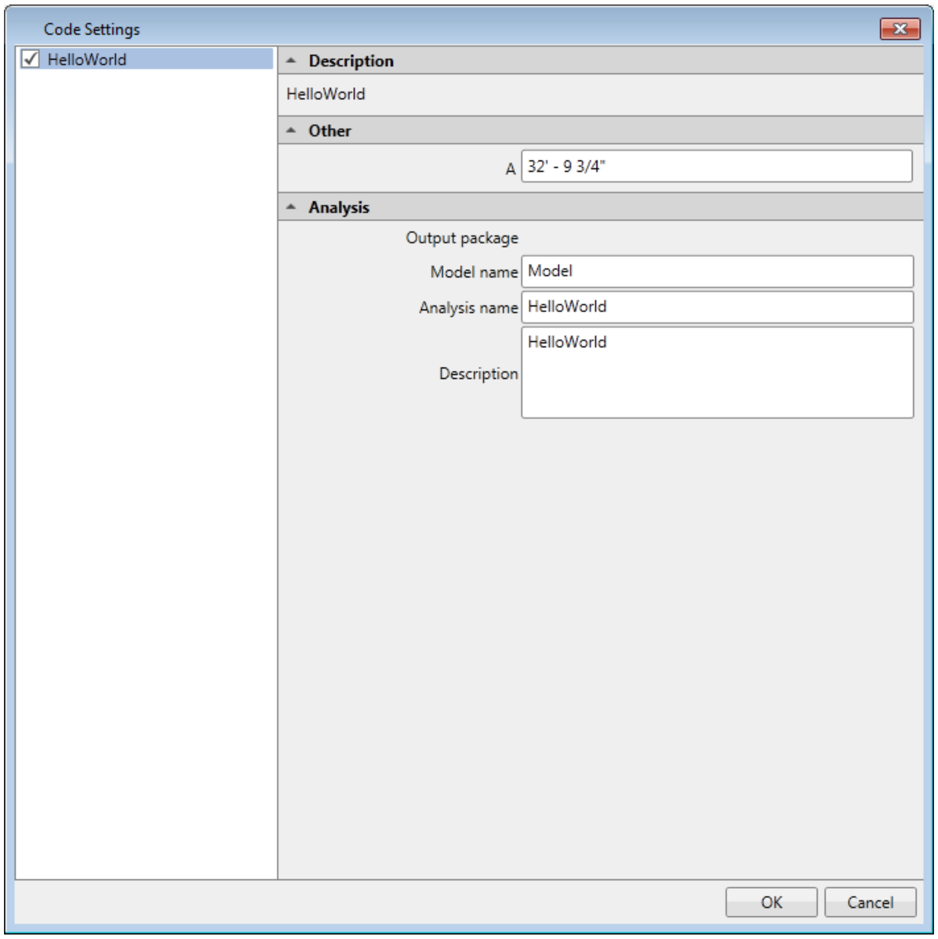
5.3. Code Checking Application installation

Using default wizard configuration, Revit addin files should be copied here:
C:\Users\<current user>\AppData\Roaming\Autodesk\Revit\Addins\2015\HelloWorldDB.addin
C:\Users\<current user>\AppData\Roaming\Autodesk\Revit\Addins\2015\HelloWorldUI.addin
Otherwise, copy these files from the Visual Studio project location inside this folder.

6. RUN HELLO WORLD CODE CHECKING

To run this new application:

- Launch Revit
- Go to Analyze tab and Select Code Settings
- The Hello World code is now Available



✓ HelloWorld

Description

HelloWorld

Other

A: 32' - 9 3/4"

Analysis

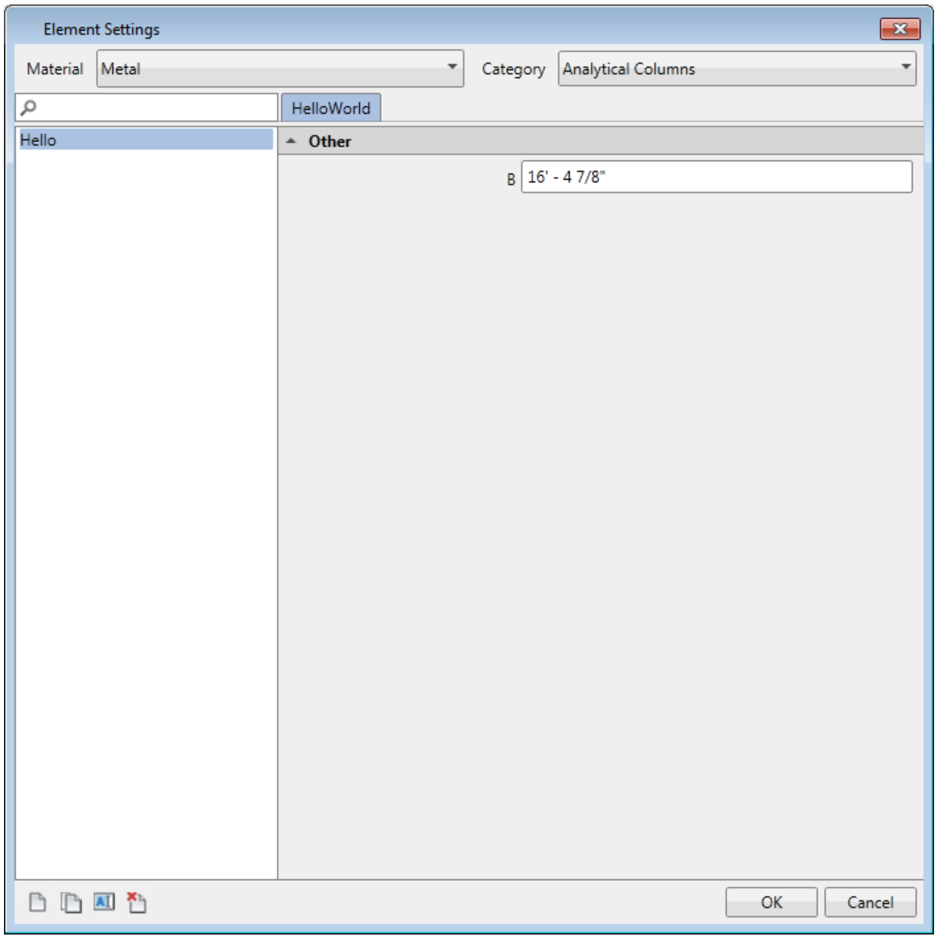
Output package

Model name: Model

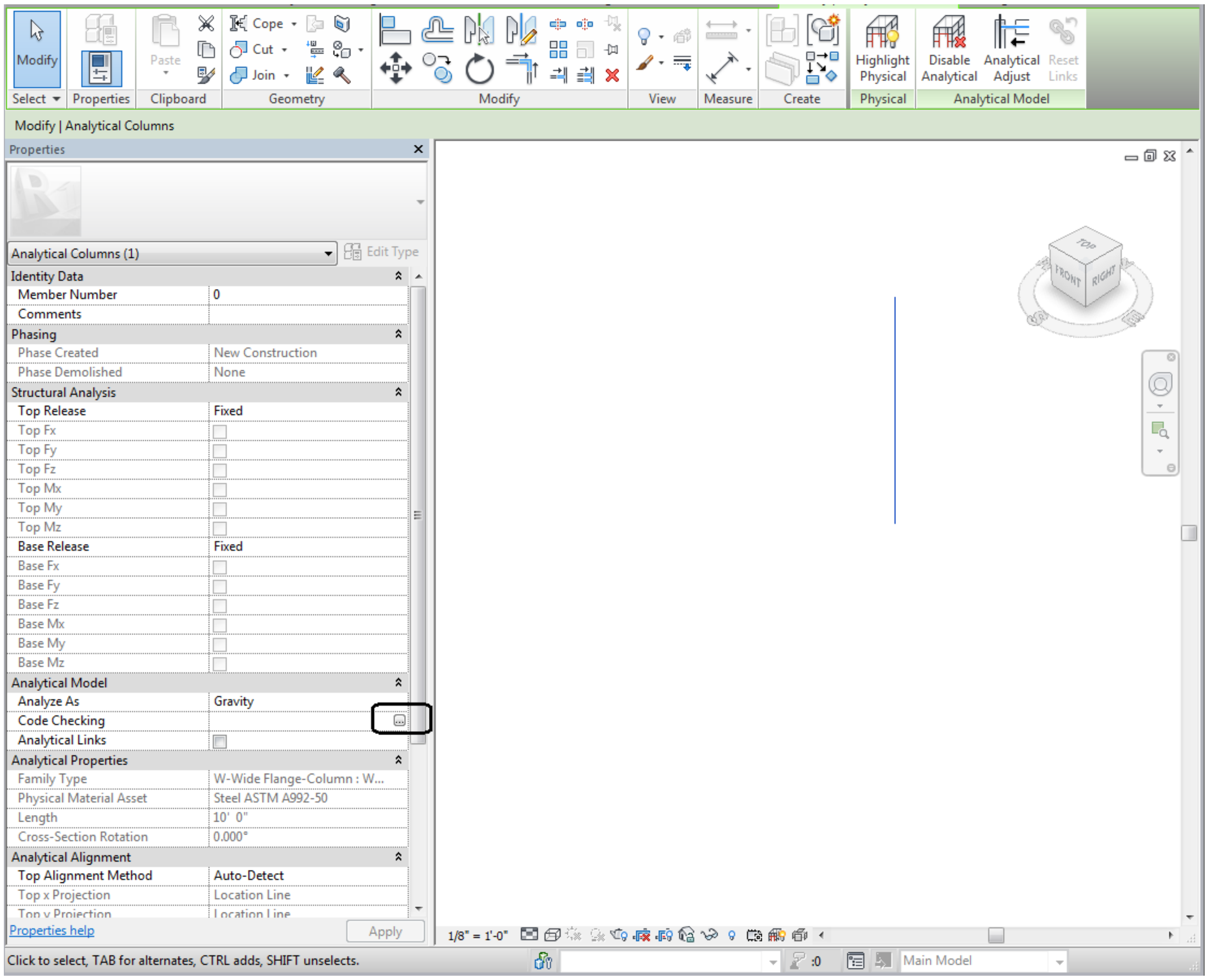
Analysis name: HelloWorld

Description: HelloWorld

- Go to Analyze tab and Select Element Settings
- Create a new “Element Settings” or rename existing one



- Create a structural column and select the analytical model
- Assign the element setting to selected analytical model by clicking on the button available inside the Code Checking parameter cell

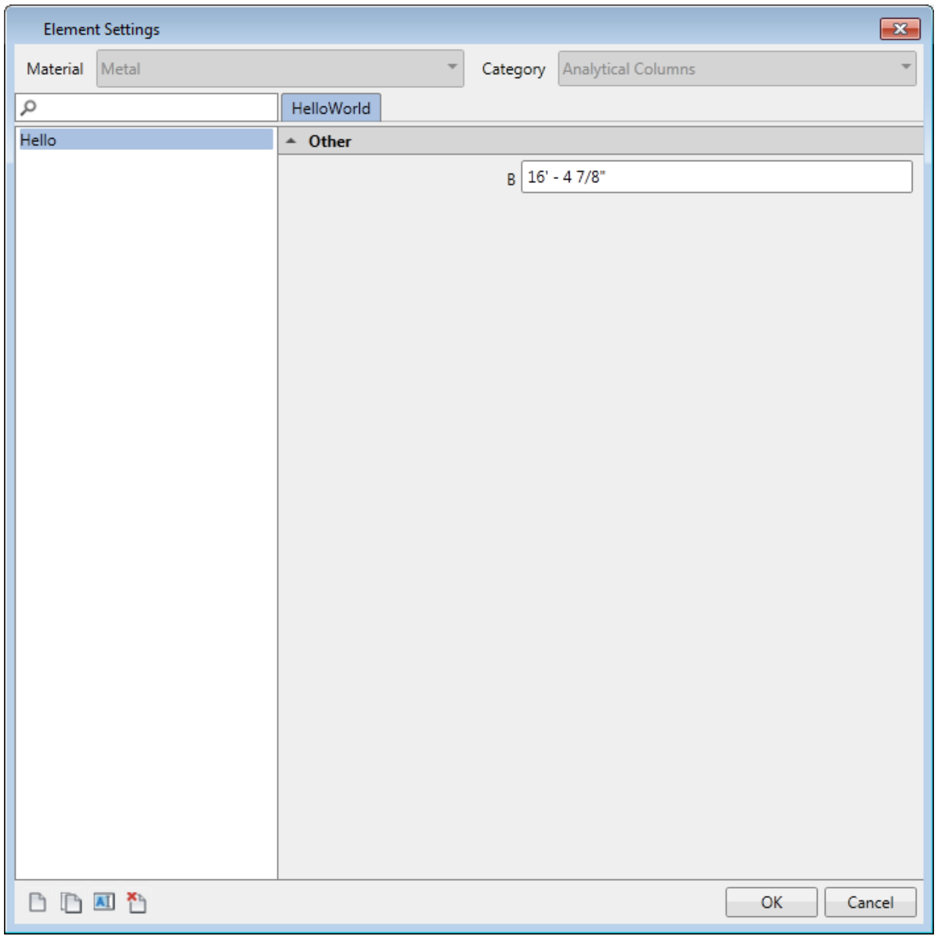


Analytical Columns (1)

Identity Data	
Member Number	0
Comments	
Phasing	
Phase Created	New Construction
Phase Demolished	None
Structural Analysis	
Top Release	Fixed
Top Fx	☐
Top Fy	☐
Top Fz	☐
Top Mx	☐
Top My	☐
Top Mz	☐
Base Release	Fixed
Base Fx	☐
Base Fy	☐
Base Fz	☐
Base Mx	☐
Base My	☐
Base Mz	☐
Analytical Model	
Analyze As	Gravity
Code Checking	...
Analytical Links	☐
Analytical Properties	
Family Type	W-Wide Flange-Column : W...
Physical Material Asset	Steel ASTM A992-50
Length	10' 0"
Cross-Section Rotation	0.000°
Analytical Alignment	
Top Alignment Method	Auto-Detect
Top x Projection	Location Line
Top y Projection	Location Line

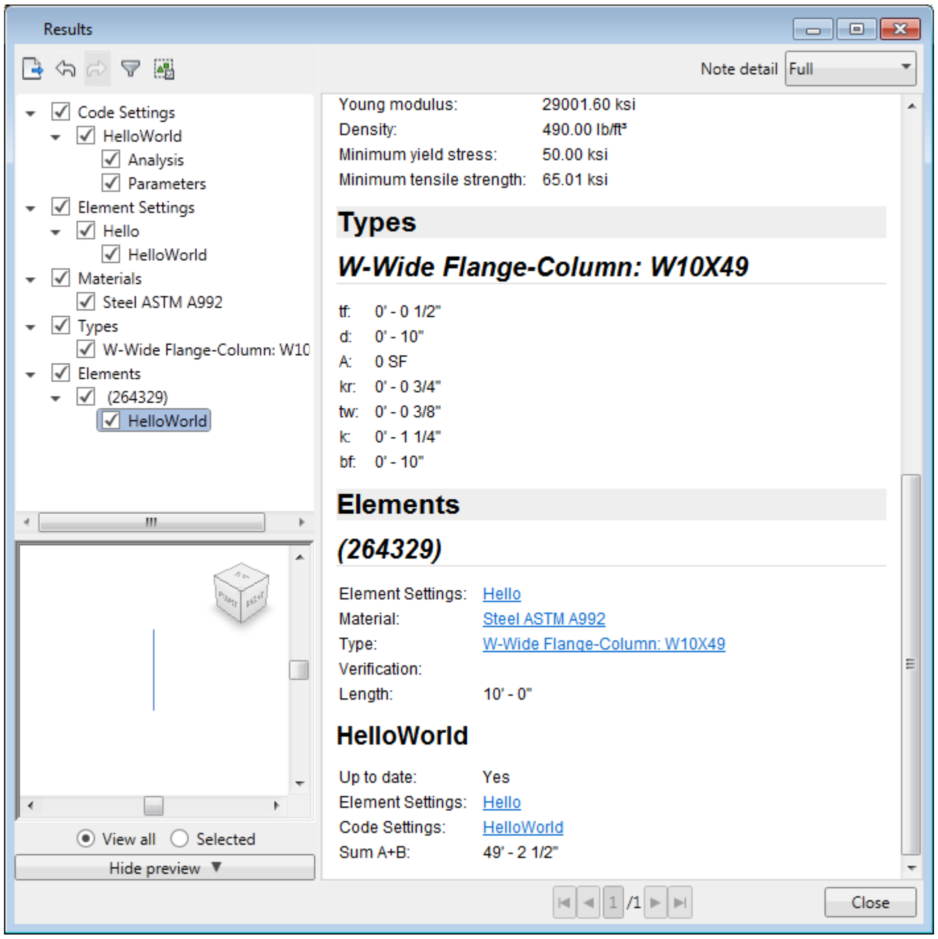
Click to select, TAB for alternates, CTRL adds, SHIFT unselects.

- When the dialog is loaded, select “Hello” and click OK



Material: Metal
Category: Analytical Columns
Hello
B 16' - 4 7/8"

- Select the analytical column element if the selection was lost
- Click on the Tab Analyze and Run Calculation
- Following dialog should appear with the results of this simple implementation



Young modulus:	29001.60 ksi
Density:	490.00 lb/ft³
Minimum yield stress:	50.00 ksi
Minimum tensile strength:	65.01 ksi

Types

W-Wide Flange-Column: W10X49

tf:	0' - 0 1/2"
d:	0' - 10"
A:	0 SF
kr:	0' - 0 3/4"
tw:	0' - 0 3/8"
k:	0' - 1 1/4"
bf:	0' - 10"

Elements

(264329)

Element Settings:	Hello
Material:	Steel ASTM A992
Type:	W-Wide Flange-Column: W10X49
Verification:	
Length:	10' - 0"

HelloWorld

Up to date:	Yes
Element Settings:	Hello
Code Settings:	HelloWorld
Sum A+B:	49' - 2 1/2"