

# TABLE OF CONTENTS

<b>1. INTRODUCTION.....</b>	<b>2</b>
1.1. EXTENSIONS STANDARDIZATION .....	2
1.2. DIALOG BOX CONCEPT .....	2
<b>2. GRAPHICAL USER INTERFACE (GUI) .....</b>	<b>2</b>
2.1. TOP 10 RULES.....	2
2.2. MAIN CONTROL .....	2
2.3. WHICH INTERFACE TO CHOOSE .....	4
2.4. EXTENSION NAME AND DIALOG TITLE.....	7
2.5. DIALOG ICON.....	8
2.6. EXTENSION AND RIBBON.....	8
2.7. FONT .....	8
2.8. SKETCH .....	8
2.9. MAIN CONTROL LIST ICONS.....	10
2.10. HELP.....	11
2.11. ABOUT .....	12
<b>3. COMMON VISUAL STUDIO CONTROLS .....</b>	<b>12</b>
3.1. RADIO-BUTTONS .....	12
3.2. CHECK BOXES .....	13
3.3. COMMAND BUTTONS .....	14
3.4. TEXT BOXES .....	15
3.5. TABS .....	15
3.6. GROUP BOXES .....	15
3.7. LIST BOXES.....	16
3.8. COMBO BOXES.....	16
3.9. LABELS .....	16
3.10. TOOLTIPS - INFO TIPS .....	17
3.11. MESSAGE BOX.....	18
<b>4. EXTENSIONS CONTROLS.....</b>	<b>18</b>
4.1. MENU .....	18
4.1.1. Menu .....	18
4.1.2. Extensions Menu .....	19
4.1.3. Extensions ToolMenu .....	20
4.2. COMBOBOXIMAGE.....	20
4.3. REXEDITBOX .....	21
4.4. REPORT GENERATOR .....	22
4.5. PROGRESS BAR.....	23
4.6. MESSAGE LIST .....	23

# 1. INTRODUCTION

## 1.1. Extensions standardization

The goal of this Design Guidelines is establishing a high quality and consistency baseline for Extensions Design.

It's important to remember that the first contact between your software and the user is the graphical interface. The design of this Graphical User interface is not all but is a base component of the software success.

Use the Extensions standard:

- Avoid custom user interface and keep a consistency between Extensions
- Reduce the time for user to understand how to use a new Extension
- Will help developer to define some explicit input data
- Will help developer to control validity of user input data and prevent error.

## 1.2. Dialog box concept

[Dialog boxes](#) are the most fundamental form of user communication.

A dialog box consists of a form filled by various control, [commit](#) buttons and a title and used to execute a task.

An explicit and well-designed dialog box makes interaction more effective between user and the software.

# 2. GRAPHICAL USER INTERFACE (GUI)

## 2.1. Top 10 rules

The most common rules are:

- Controls must be [aligned](#)
- [Standard distances](#) between controls must be applied
- Controls must be [ordered](#) with a logical or a natural progression
- Controls must be [ordered](#) according to tab control selection
- [Default values](#) must be set according user needs
- All strings must be in resource files
- All images files must be in resources files
- All labels must be translated according to localization rules
- Maximum size of dialog is 1024 x 768 pixels.
- Follow [Windows User Experience Interaction Guidelines](#) and [Revit API User Interfaces guidelines](#)

## 2.2. Main Control

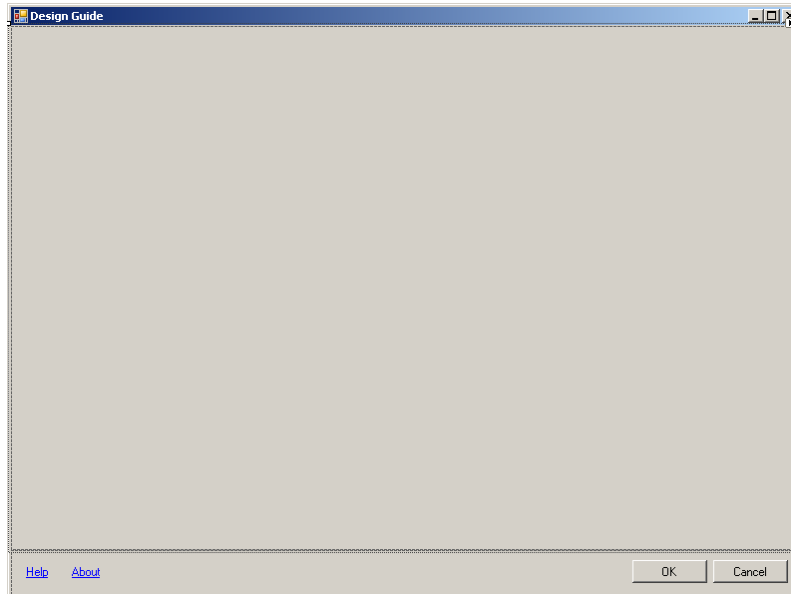
To keep a constant look for each Extension, Extensions template contains automation for it.

The standard functionality of template gives possibility to embed user controls in Main Control. Main Control is automatically embedded in main module window.

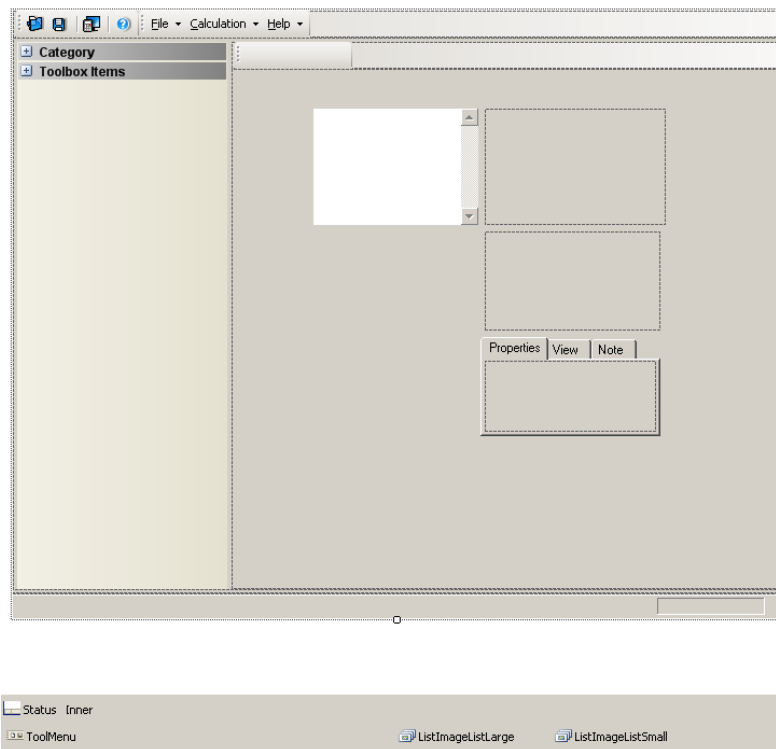
The order of designing standard module looks as follows:

Specify all options accessible in left panel, additionally divided in groups.  
Design all user controls in connection with particular options.

The main form is a container for all user controls.



Main control is a template.



This control contains most common controls to manage menu, toolbar and tool menu, tab control, web browser, images list, toolbox and host toolbox.

This main control can be customized during the wizard generation or configured by flags.

Flag description:

- List – visibility of left option panel
- Tab Dialog – user control visibility in right panel. User control is visible on a tab
- Tab View – visibility of control view (like a 2D view) or a user control with another user interface in right panel. User control is visible on a tab
- Tab Note – visibility of control with an HTML note in right panel (embedded Web Browser). User control is visible on a tab.

Using only tabs flags cause controls not to be embedded in Tab Control container but directly positioned in right panel without tabs.

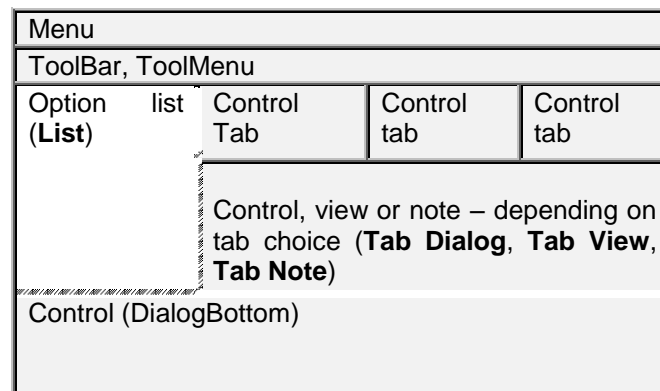
- Dialog Bottom – visibility of additional user control located on the bottom of the dialog
- Status – status bar visibility
- Menu – menu bar visibility
- Toolbar – toolbar visibility
- Toolbar Inner – right panel toolbar visibility
- Toolbar Menu – visibility of toolbar with docked menu
- Activate Control – activation of a dialog control, view or note after option choice (not implemented in current version)
- Form Fixed – determine if dialog has a constant size or if user can change window size
- VSplitFixed – vertical split bar
- HSplitFixed – horizontal split bar
- Disable Docking – toolbar docking inside window
- ToolbarIncrementForm – increase height of a main window (Main Form) by menu bar height or toolbar height. It's useful when changing window look from toolbar menu to menu bar
- AutoSizeForm – automatic window size adjustment to cover all used controls, by default it's necessary to set main window size (Main Form) manually to cover main control (Main Control) and user controls (not implemented in current version).

## 2.3. Which interface to choose

The look can be constant or different according to designer choice.

Following GUI configurations are possible:

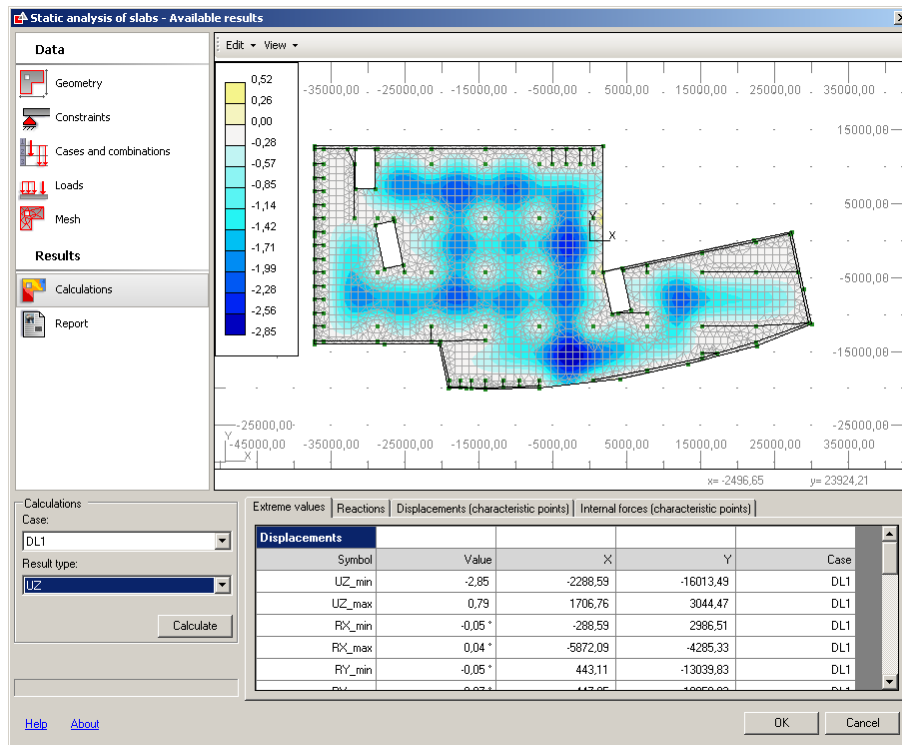
- Complete dialog configuration



Status

- Using single Tab flag and bottom side dialog.

ToolBar, ToolMenu	
Option list (List)	Control, view or note ( <b>Tab Dialog</b> , <b>Tab View</b> , <b>Tab Note</b> )
Control (DialogBottom)	
Status	



Reinforcement of beams

File Help

Geometry

Storage

Storage distribution

Addition top bars

Addition bottom bars

Divide bars

Reinforcement areas

Layouts list

Identification

Family name: RC\_Concrete Rectangular Beams

Type: 400 x 600mm

Geometric parameters

Number of spans: 1

L: 6000

b: 400

h: 600

L1: 600/2

h1: 0

Design parameters

Level: 3rd floor

Offset: 0

Offset: 0

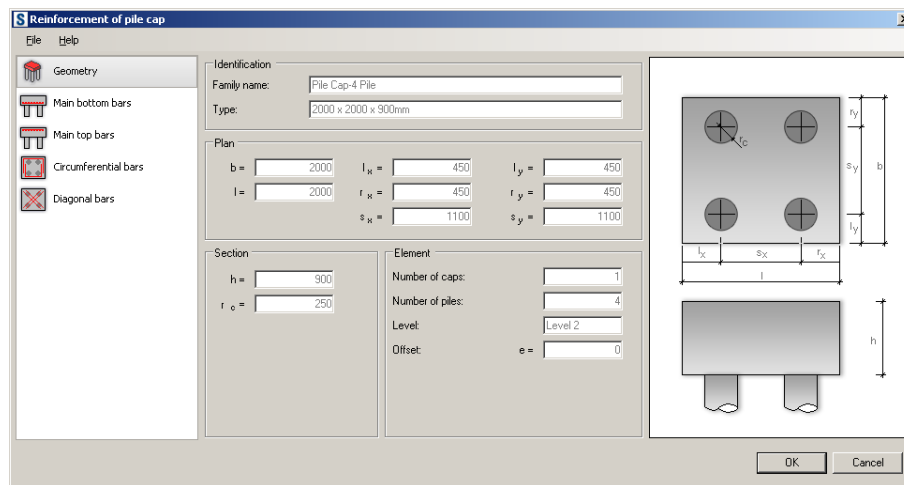
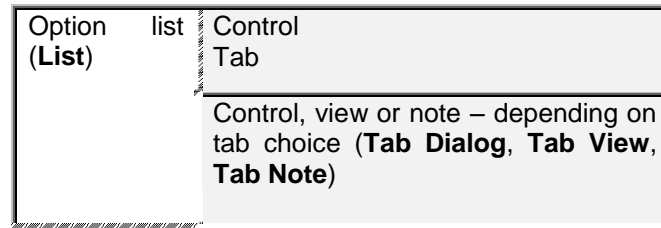
Without reinforcement generation

User control

OK Cancel

The size recommended for this dialog configuration is 900 x 750 pixels.

- Using one tab flags without bottom side dialog. List | Tab Dialog | Tab View | Tab Note or different tab flags combinations.

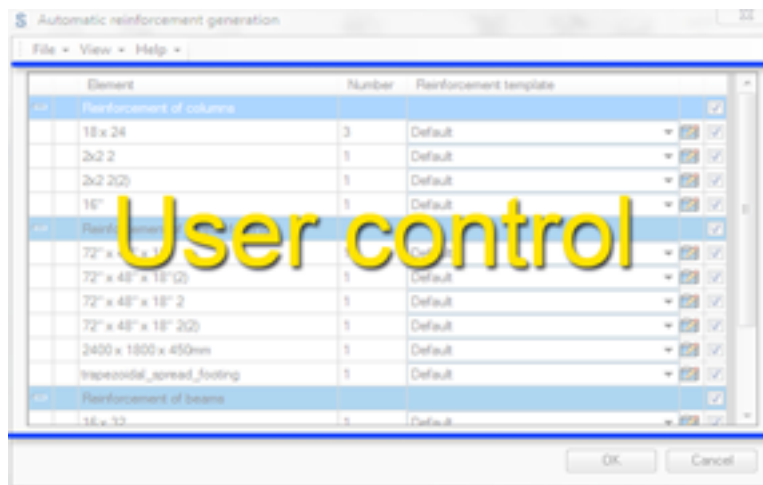
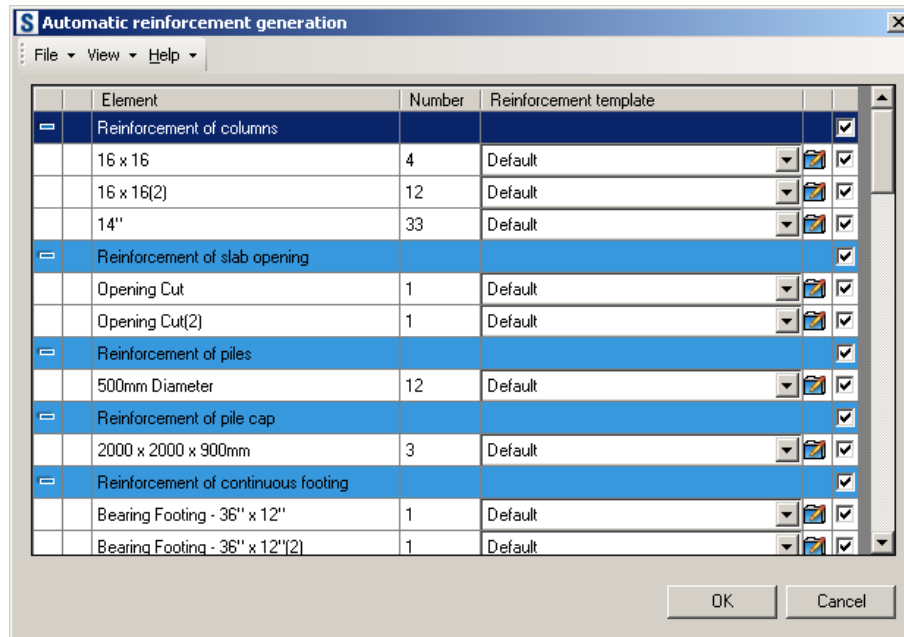


The size recommended for this dialog configuration is 900 x 500 pixels.

- Using one tab flag to design a simple dialog.

ToolBar, ToolMenu

Control, view or note – depending on tab choice (**Tab Dialog**, **Tab View**, **Tab Note**)



The size recommended for this dialog configuration is 800 x 450 pixels.

The classes REXUI and REXLayout are responsible for setting window look and for interactions with options..

## 2.4. Extension name and dialog title

The name of extension is composed of 2 names: a short one for the Revit extension manager and a long one for the main dialog title.

These names must be user friendly and in relation with the functionality.

The long name must be present on each dialog and must be constant.  
This long name could be used present a tooltip on the Revit Ribbon.

## 2.5. Dialog Icon

All extensions must have an icon on the left top corner.

The size of this icon must be 32x32 pixels.  
The default icon is REX.ico.  
Developer could replace REX.ico by his own icon

## 2.6. Extension and Ribbon

Each extension could be launched as an external command directly from the addin list, could create his External application and hosted some extensions on a specific panels (and tab).

An icon must be placed on the directory configuration of extension with "ExtensionName.png" as a name and be created in png format 32x32 pixels to support the ribbon integration. A second file named "ExtensionName\_small.png" is welcomed to support the quick access toolbars. If this second file is not present, the 32x32 file will be resized dynamically.

For more information regarding icon, please refer to the [Autodesk Icon Guidelines](#).

## 2.7. Font

The font to use for all control and label is the default system font defined in Visual Studio 2010 - Microsoft Sans Serif, 8.25pt.

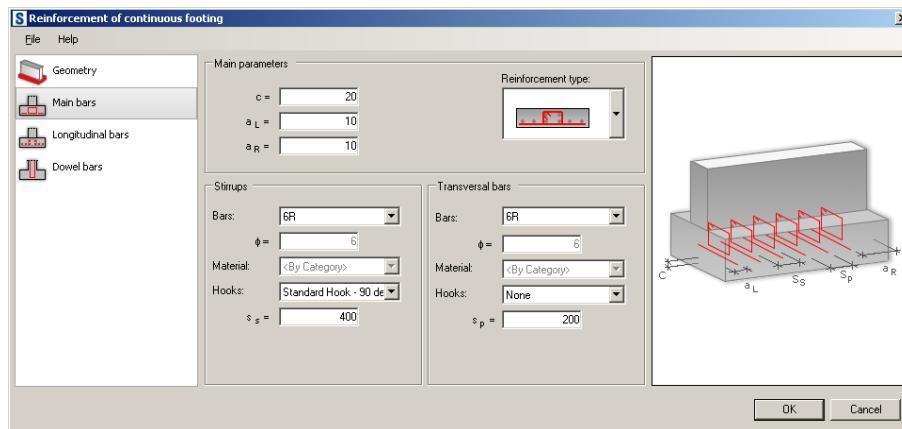
## 2.8. Sketch

The aim of a sketch (included on a box control picture) is to provide some friendly information about a control to user to help him understand how to set data.

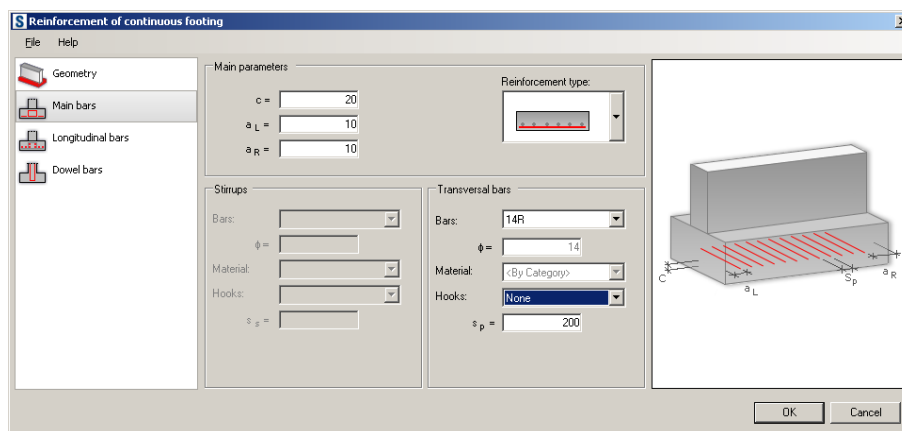
If a sketch is displayed, all symbols defined on the dialog must be present on this sketch.  
Fonts to use are "Ms Reference sans Serif" or "Arial".

A dynamical management of this sketch, according to the context and data available, is recommended.



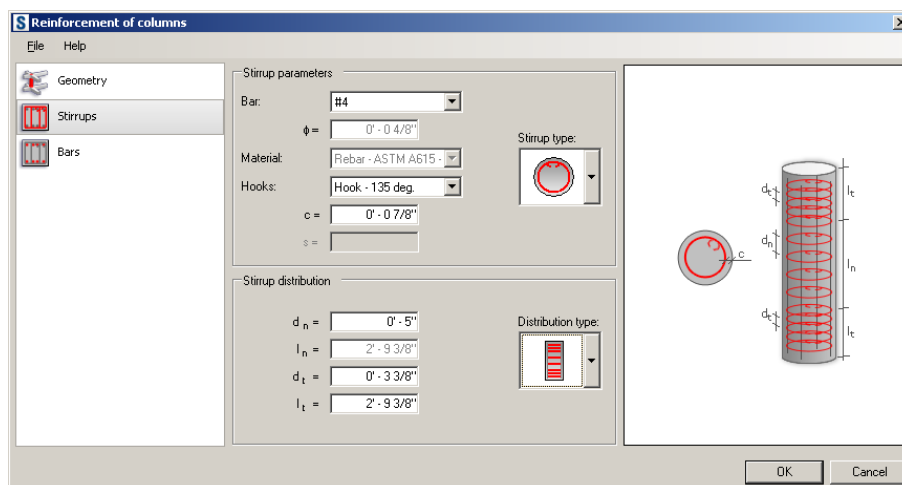


Continuous footing with stirrups



Continuous footing without stirrups

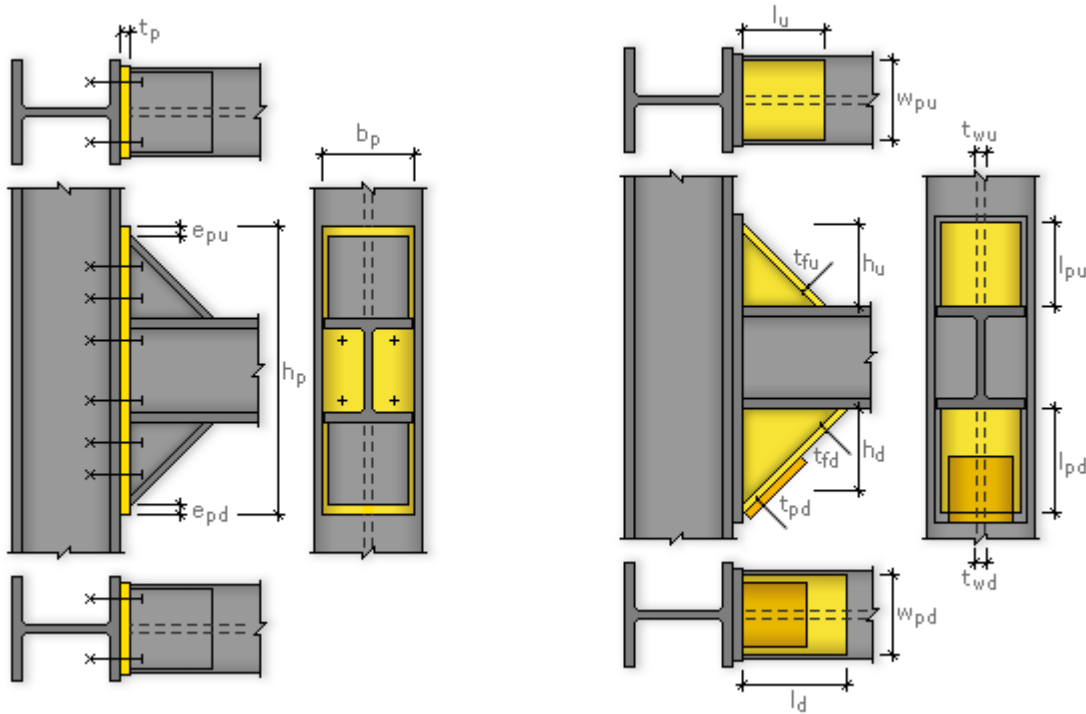
Generally, the idea in Extensions is to display this drawing in 3D but sometimes some 2D drawings are welcomed to give some additional information or if a 3D drawing is not clear.



For reinforcement extensions, the color codification is to have the concrete element in grey (RGB 192, 192, 192), rebar to define in red (RGB 255, 0, 0), and dimension lines in dark grey color (RGB 81, 81, 81).

On a group of extensions, drawings must use common rules.

For steel connections, steel elements are in grey, element to define in yellow, and dimension lines in black.

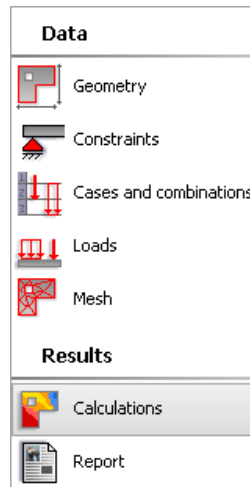


It is important to use a graphical expert to make drawings.

## 2.9. Main control list icons

As explained above, the main control could have a list. On this list image format must be 32x32 pixels. And follow the [Autodesk Icon Design Guidelines](#).

It's important to have some distinguishable icons and to choose them according functionalities linked.



## 2.10.Help

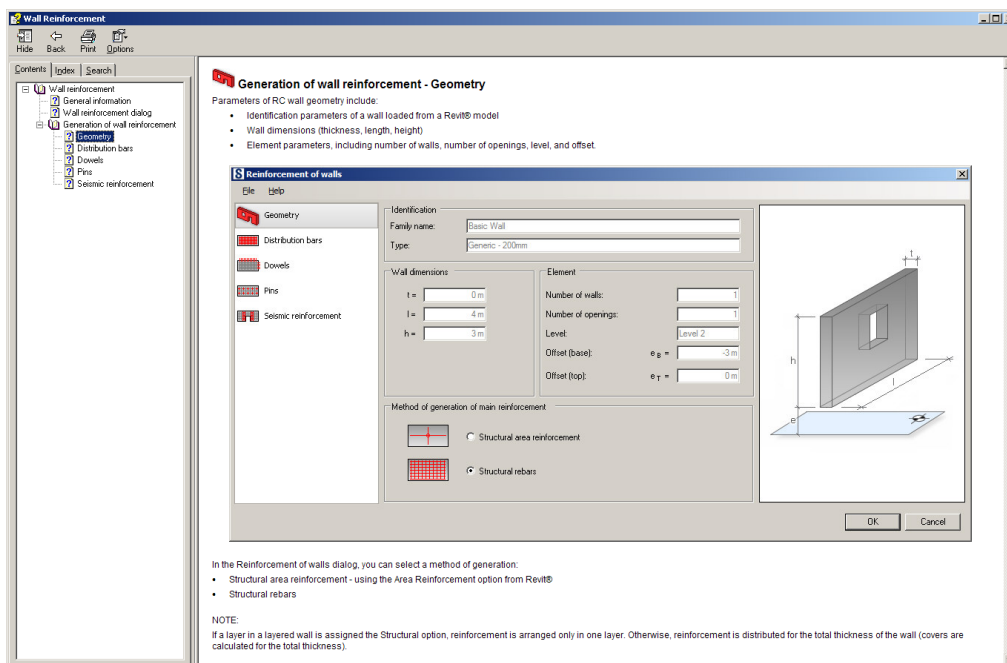
All extensions must have CHM help file format associated and accessible by F1 keyboard short cut.

The goal of a Help system is to anticipate every question that every user might have during Extensions using.

The content of the help file is designed to assist users when they are unable to complete a task, want to understand a concept in more detail, or need more technical details than are available in the UI. The content must be focused on the important scenario and avoid obvious content.

The limitation of Extension must be added on the help system as a note.

The Help system must have different entry points like a table of content, an index and a search system.



## 2.11.About

All extensions must have an About dialog box to provide some general Extension information such as version, licensing, copyright and Extensions Engine version.

## 3. COMMON VISUAL STUDIO CONTROLS

Common controls used on Extensions are:

- Radio buttons
- Check boxes
- Command button
- Combo boxes.

The Extensions common rule is to use for all control default value for each property define in Microsoft Visual Studio.

Please refer to Microsoft UX guidelines.

### 3.1. Radio-Buttons

This [control](#) allows user a choice among a set of mutually exclusive options.

This control could be used when the number of option is between 2 and 7 and you would like to draw attention to the options.

In case of 2 options, a single checkbox is preferred in Extensions to turn off/on an option.  
For more than 7 options, drop-down list must be used.

All radio button groups must have a label. This label must be explicit and not redundant with radio-label text.

The list options should be a label and be organized on a logical order. The first value must be selected as default value.

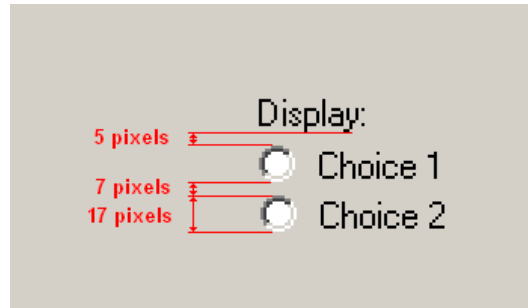
Alphabetical ordering is not recommended according to language dependences.

Vertical alignment of radio button is easier to read than a horizontal one and eliminate some localization issues.

Radio button label must start in upper case and be ended without any punctuation. In a case of multi-line labels (used as subordinate control) the top of the label must be aligned with the radio-button.

The recommended sizing and spacing for radio buttons are:

- Height: 17 pixels
- Spacing between option: 7 pixels
- Margin: 3 pixels.



## 3.2. Check Boxes

This [control](#) allows user a choice between opposite choices.

This control is used when the choice is not ambiguous and mainly to turn on/off an option or select/deselect an element. If it is not easy for user to understand what the opposite option is, radio buttons could be a better choice.

This control could be used alone or in a group

A single check-box is used to make an individual choice. A group of check boxes is used to make zero or more choices.

For more than 10 options, use a checkbox list.

A list of check-boxes should have a label and be organized in a logical order or with a natural progression.

Vertical alignment of check box is easier to read than a vertical one and eliminate some localization issues.

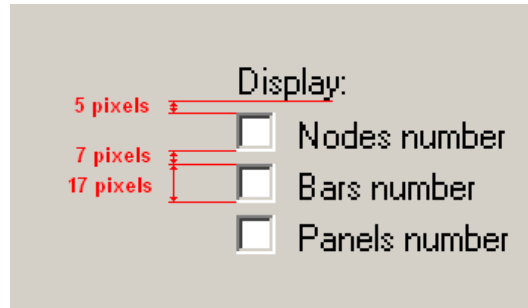
Check box label must start in upper case and be ended without any punctuation.

In case of subordinate controls like textbox or group of control, place single control on the right and group of control below the checkbox and try as much as possible to align labels with check box label.

All Check box groups must have a label. This label must be explicit and not redundant with Check box label text.

The recommended sizing and spacing for check boxes are:

- Height: 17 pixels
- Spacing between option: 7 pixels
- Margin: 5 pixels



### 3.3. Command Buttons

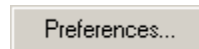
This [control](#) allows user to execute an action immediately. If the action is not immediate, then you need to display a progress bar.

In Extensions, generic command buttons (OK, Cancel) are recommended.

Sometimes according to question, the YES, NO generic command button could be use.

To indicate that command needs more information to be executed, you could add ellipsis at the end of the button label.

In Extensions, if this kind of button is needed perhaps you need to create a new layout to fill data.



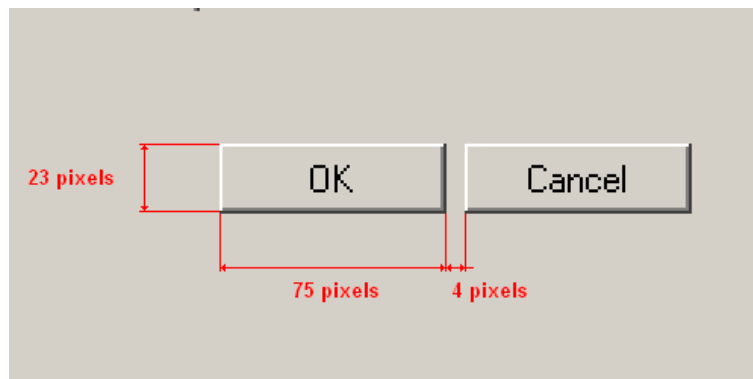
Every button must have a label or graphics. Label must start in upper case and be ended without any punctuation.

In case of a text label, text must start by an imperative verb and could contain a noun. Some exceptions are acceptable for Advanced, Back, Details, Forward, Less, More, New, Next, No, OK, Options, Previous, Properties, Settings, and Yes. Note that Advanced is reserved for advanced users with a specific knowledge. For directional button, labels are '<->' and '<->'. For directional button, labels are '<->' and '<->'.

If the command button appears more than once in one dialog, try using the same label text and window location.

The recommended sizing and spacing for buttons are:

- Size: 75 x 23 pixels
- Spacing between: 4 pixels.



This size is the minimal one for a button with labels. In case of command button with graphic, the size of this button could be adjusted to the graphics.

On Extensions common rules, OK means a validation of the data and an action on Revit, Cancel is equivalent to the key "Esc". In this last case user goes back to Revit without any particular action.

### 3.4. Text Boxes

Use text box control provide with Extensions SDK in this case.

### 3.5. Tabs

This [control](#) is a way to display to the user relationship on a set of controls and to increase the window surface area to show related information on different pages.

If this kind of control is needed, two ways are possible:

Use the list control and create a new layout to avoid a new dimension on the extension.

Use control tab with the flag configuration feature.

If the choice was done to use tab control, all tabs must have label.

Label must start in upper case and be ended without any punctuation and be placed as heading.

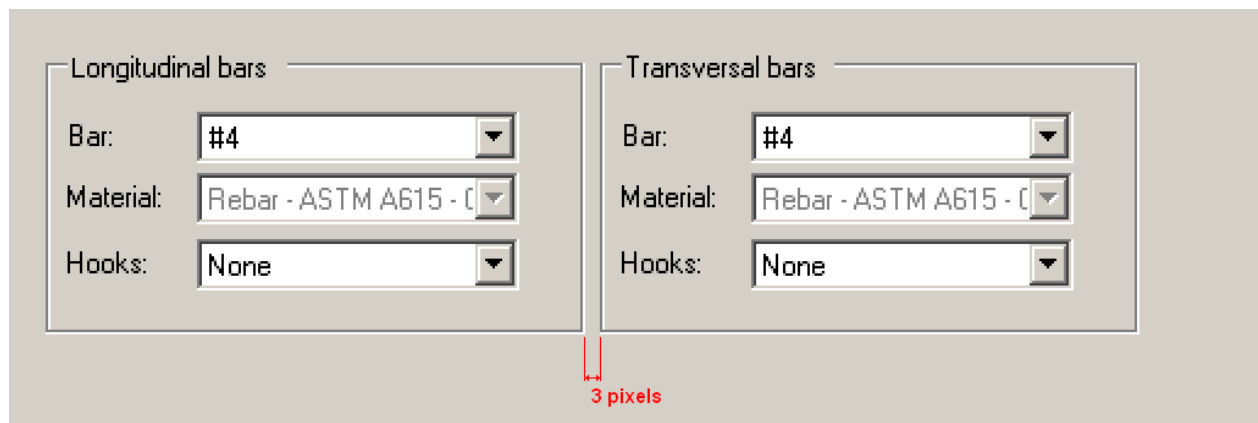
### 3.6. Group Boxes

This [control](#) is a way to display to the user relationship on a set of controls.

This control is used if we have more than one control on the group and the relation between controls is well established. In case of one control, we use a classical label.

All group boxes must have a label, and must start in upper case and be ended without any punctuation.

In case of multi group boxes in a window, use the similar phrasing for each one.



The common rules in Extensions are:

- Do not use group of group
- Do not put any control on group labels

- Never disable the group box itself but controls inside
- Combo boxes label are aligned with group box label
- The common and constant spacing between groups is 3 pixels.

## 3.7. List boxes

In Extensions development rules, we try to find another solution than to use List boxes.

If necessary, use the [common Microsoft rules](#).

## 3.8. Combo Boxes

[Combo box](#) allows user to select a unique choice on a predefined list of option.

In Extensions, we use Drop-down List.

This control is used if we want to enumerate choices. Typically in Extensions, some data read on the current Revit project.

All Drop-Down Combo boxes must have a label, and must start in upper case and be ended with ‘:’

In general, we do not use this control to perform a control or to display other windows.

The Drop-Down Combo boxes should have a label and be organized in a logical order. The first value must be selected as default value.

Alphabetical ordering is not recommended according to language dependences.

In general it is useful to introduce an option like all and/or none at the beginning of the list if necessary.

The recommended sizing and spacing for Combo box is:

- Height: 23 pixels
- Distance to label: 5 pixels
- Spacing: 7 pixels



The width is linked to the longest valid data – Drop-down list cannot be scroll horizontally.

For localization issues, it is recommended to use the length of the longest valid data and include an additional 30 percent.

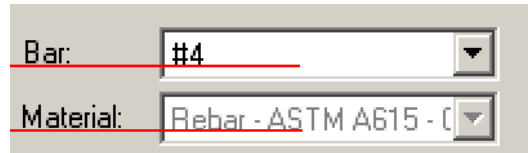
## 3.9. Labels

A label gives some information to the user about one specific control.

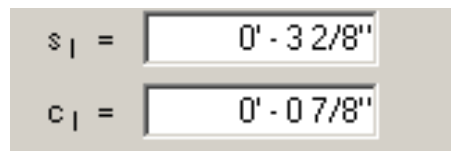


Every edit box and combo box must have a label to describe data to input or select on a specific control.

On the common Extensions rules, labels for REXEditBox and ComboBox, are to place them on the left side of the control and to align the text with text include on the control to describe.



To define a symbol, linked with a drawing, the common rule is to write one or more letters, subscript text and the = sign on 3 different labels.



To define it, the subscript, the common rule is to use a smaller font size, 7pt.

All labels must have the same spacing between, symbol and subscript part, subscript part and equal sign, and equal sign and REXEditBox.

The equal sign must be at 5 pixels of the control; the height of the label is 13 pixels.

The “=” sign must be aligned on the right.

To define a label for a ComboBox the common rule is to write a text, with an upper case as first letter, aligned with the group box label and with ‘:’ at the end.

It’s important to anticipate the translation issues and to keep enough places for this translated label – especially for German language.

The “:” sign must be aligning on the left.

An exception could be done when the unit is diameter, we could put the label in the right part with the same rules for spacing and position as describe upper.



## 3.10.Tooltips - Info Tips

A [tooltip](#) is a small pop-up window that labels the unlabeled controls.

This control is used mainly for unlabeled toolbar controls or command buttons and to give an additional value for the final user.

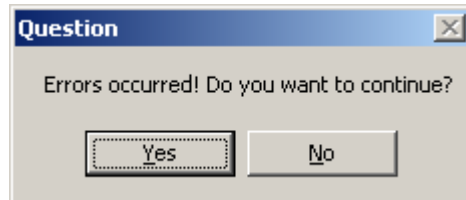
In Extensions, this control is used only in the Revit Manager and for the toolbars.

The common rule is not to use this kind of control. The User Interface according rules defined in this document must be clear and all controls must be labeled.

If this control is really needed, the main issue is to know which useful information provide to user. The rules are to start with an upper case letter and to end without any punctuation.

## 3.11. Message box

A Message Box notifies some events to user and requires an immediate action.



Don't abuse of Message Box, this mechanism is not user friendly.

The text displayed in a message box must include only one piece of information. If more, you need to use an Extensions message list.

Message box are mainly used to recognize and identify errors.

Usage of Revit APIUI task dialog is also a recommended solutions.

## 4. EXTENSIONS CONTROLS

Extensions SDK gives the ability to use some controls. On the rest of this document are methods to use them.

### 4.1. Menu

A menu is a list of commands available to users in the current context.

In Extensions, two kind of menus could be used, Extensions Menu (Menu bars) and Extensions ToolMenu (Toolbars Menu).

#### 4.1.1. Menu

Each menu item is an individual command.

Menus are displayed from a menu bar on the top left of the main control.

Menu are used for immediate actions, however several information could be needed. In this case, we could use ellipsis at the end of the label.

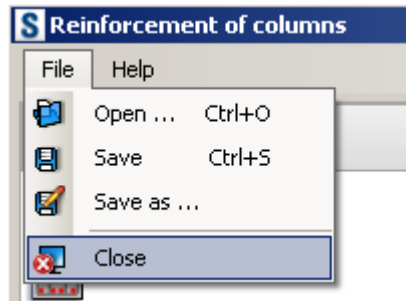
Don't use ellipsis for help and about.

The label of a menu starts always by an upper case.

A submenu could be use to create a hierarchy of commands and is indicated by an arrow at the end of the submenu label.

### 4.1.2. Extensions Menu

A Menu bar is used to display commands in different part of the extension.



The standard menu bar structure is as follows. This list shows the menu category and item labels, their order with separators, their access and shortcut keys, and their ellipses.

<u>F</u> ile		
<u>N</u> ew	Ctrl+N	
<u>O</u> pen...	Ctrl+O	
<u>C</u> lose		
<separator>		
<u>S</u> ave	Ctrl+S	
Save <u>a</u> s...		
<separator>		
Se <u>n</u> d to		
<separator>		
<u>P</u> rint...	Ctrl+P	
Print preview		
Page setup		
<separator>		
<u>1</u> <filename>		
<u>2</u> <filename>		
<u>3</u> <filename>		
...		
<separator>		
<u>E</u> xit	Alt+F4	(shortcut usually not given)
<u>E</u> dit		
<u>U</u> ndo	Ctrl+Z	
<u>R</u> edo	Ctrl+Y	
<separator>		
<u>C</u> ut	Ctrl+X	
<u>C</u> opy	Ctrl+C	
<u>P</u> aste	Ctrl+V	
<separator>		
S <u>e</u> lect a <u>l</u> l	Ctrl+A	
<separator>		
<u>D</u> elete	Del	(shortcut usually not given)
<separator>		
<u>F</u> ind...	Ctrl+F	
Find next	F3	(command usually not given)

Replace...      Ctrl+H  
Go to...        Ctrl+G

#### View

Toolbars  
Status bar  
<separator>  
Zoom  
Zoom in        Ctrl++  
Zoom out      Ctrl+-  
<separator>  
Full screen    F11  
Refresh        F5

#### Tools

...  
<separator>  
Options

#### Help

<program name> help F1  
<separator>  
About <program name>

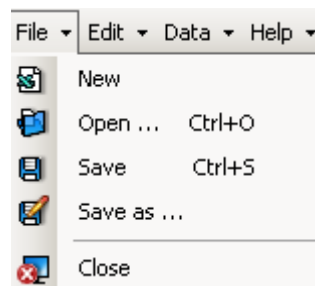
### 4.1.3. Extensions ToolMenu

Tool bar menu is used in Extensions when interfaces don not have any lists and are composed of only one dialog.

Toolbar menu could be used if it's necessary to introduce a menu management on a sub control or on a particular tab. This approach must be justified by a real final user need.

In toolbar menu, the main idea is to display only few commands.

In the toolbar menu and for each item an icon could be placed on the right part or have the ability to be checked.



## 4.2. ComboBoxImage

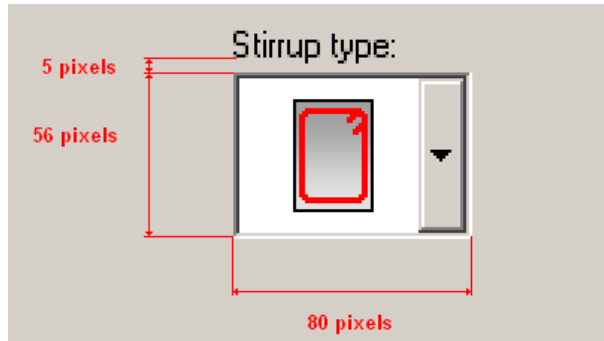
ComboBoxImage allows user to select a unique choice on a predefined list of options. This list is composed in this case by images

Combo boxes Image must have a label at the top, aligned on the left, and must start in upper case and be ended with ‘:’

Image must be organized in a logical order. The first value must be selected as default value.

The recommended sizing and spacing for Combo box are:

- Width: 80 pixels
- Height: 56 pixels
- Distance to label: 5 pixels



### 4.3. REXEditBox

REXEditBox allows user to define a decimal or a string value.

This control manages and displays unit according to the running context of the extension and the type of unit to display.

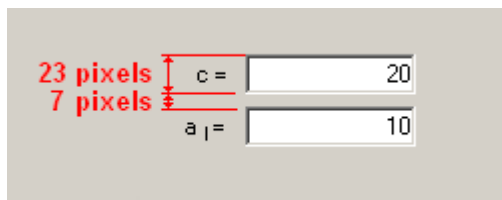
A mechanism of validation and rounding is implemented too.

REXEditBox must have a label, and must start in upper case and be ended with ‘:’  
The position of the label must be aligned with the label of the container group label.

A default value must be set.







The recommended sizing and spacing for REXEditBox is:

- Height: 23 pixels
- Distance to label: 7 pixels



The width is linked to the longest valid data for localization issues, it's recommended to use the length of the longest valid data and include an additional 30 percent.

The Visual studio default value must be used:

AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoCompleteCustomSource	<b>(Collection)</b>
AutoCompleteMode	None
AutoCompleteSource	None
BackColor	 Window
BorderStyle	Fixed3D
CausesValidation	True
CharacterCasing	Normal
ColorBackgroundDisabled	 <b>Control</b>
ColorBackgroundError	 LightCoral
ColorBackgroundOK	 LightGreen
ColorBackgroundStandard	 <b>Window</b>
ContextMenuStrip	(none)
Cursor	IBeam
DataType	<b>DECIMAL</b>
Dock	None
Enabled	True
EngineeringFormat	False
Epsilon	<b>0</b>
Font	Microsoft Sans Serif, 8.25pt
ForeColor	 WindowText
GenerateMember	True
HideSelection	True
ImeMode	NoControl
InvalidCharacters	
Lines	<b>String[] Array</b>
Location	<b>461, 15</b>
Locked	<b>True</b>
Margin	3, 3, 3, 3
MaximumSize	0, 0
MaxLength	32767
MinimumSize	0, 0
Modifiers	Private
OnDisableAction	<b>eSTANDARD</b>
RangeMax	<b>0</b>
RangeMaxCheck	False
RangeMin	<b>0</b>
RangeMinCheck	False
ReadOnly	False
RememberCorrect	<b>True</b>
RightToLeft	No
RoundingFractional	<b>ROUNDING_1</b>
RoundingIncrement	<b>0.01</b>
ScrollBars	None
Separator	.
ShortcutsEnabled	True
Size	<b>85, 20</b>
TabIndex	<b>52</b>
TabStop	True
Tag	<b>UC_StructureDim</b>
TextAlign	<b>Right</b>

## 4.4. Report Generator

Report generator (ROHTML) is a reusable component for creating report.

The aim of this control is to provide to the user a report on an HTML viewer.

This control gives the ability to save the report on mht file format. This format of files could be read by the Office Suite and include images on a unique file.

See ExtensionsFrameworkAPI.chm files for more information.

## 4.5. Progress bar

The aim of this control is to provide to the user information about the status of a not immediate action.

We use the Extensions progress bar when the action is not immediate.



It is better to use determinate progress bar to provide better feedback to user.

Do not restart progress. Users have no way of knowing when the entire operation will be complete. If it resets it won't be displayed.

Do not provide unnecessary details. A well-labeled progress bar provides sufficient information, so provide additional progress information only if users can do something with it.

## 4.6. Message list

The aim of this control is to communicate some useful information to user.

If the message could be written in one line, choose the message box, otherwise use the warning list.

The advantage of the warning is that we can have 3 categories, Errors, warning, messages.

All strings displayed must start by a symbol.

Errors are used to display some real errors. These errors are the result of not achieving the tasks.

Warning is used to display some potential errors but the current task was completed.

Message is used to give useful information to the user.

