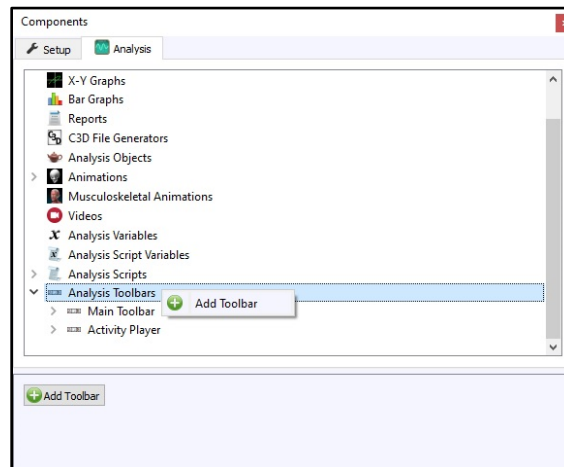


The MotionMonitor xGen Software Guide: Using Toolbars and Toolbar Buttons

This document reviews the process of using and creating Toolbars, including adding buttons to execute scripts and other Toolbar functions in The MotionMonitor xGen. Toolbars allow for customization, ensure a consistent workflow, and help simplify the operation of The MotionMonitor xGen by eliminating the need to access menu items or the Components window.

Just as with many other elements within The MotionMonitor xGen, Toolbars can be configured through the Setup and Analysis tabs in the Components window. So which tab is the best to use? Well, it depends. The setup side is generally associated with tasks pertaining to the setup of a data collection session and capturing information at the time of a collection and is unchangeable thereafter. The analysis side is looking at data in the present and can be modified in post processing. Components on the Analysis side will be overwritten when loading an Analysis. The same functionalities can be performed from with a Toolbar on the Setup side as from a Toolbar on the Analysis side. However, a Toolbar Button can only run scripts from the same side of the Components window (i.e. a Setup Toolbar cannot reference an Analysis script) and Toolbar Integer and Script Edit fields can only utilize script variables from the same side of the Components window.

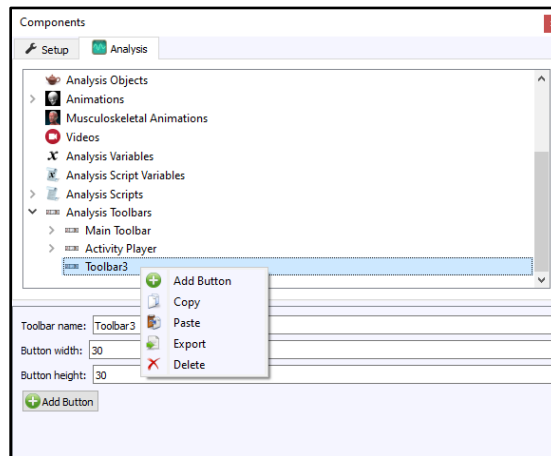
Toolbars can be added by right-clicking on the Permanent Toolbars node in the Setup Components tab or Analysis Toolbars in the Analysis Components tab and selecting the Add Toolbar option or by right-clicking on the Add Toolbar button in the parameters panel when a Toolbars node is selected.



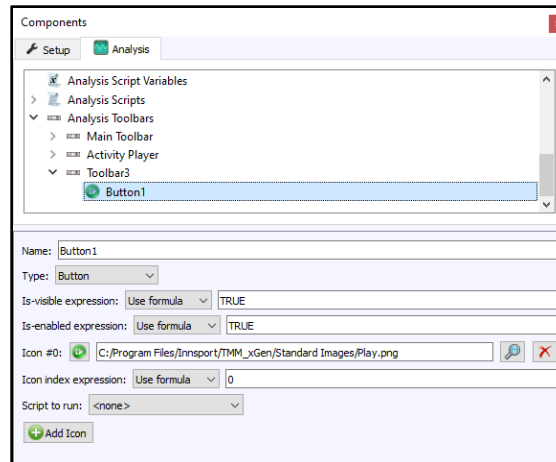
Toolbars can be undocked and repositioned in the main application's window or within Activities by clicking on dots within the Toolbar and dragging it to a desired location. The positioning of a Toolbar will be saved in Workspace and Analysis files.



To modify a Toolbar, click on the Toolbar to bring up the parameters panel. Buttons can be added by right-clicking on the Toolbars node and selecting the Add Button option or by clicking on the Add Button button in the parameters panel when a Toolbars node is selected. A Toolbar can also be copied and pasted to create a duplicate Toolbar, Exported to a Component Set or Deleted through the right-click menu. The Toolbar can be given a name, which will be displayed in the View menu to hide and show the Toolbar. The Button width and height can also be modified through the Toolbar parameters panel.



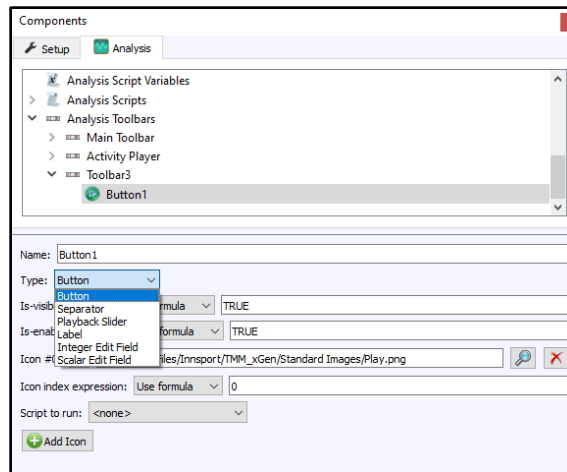
Once a Button has been added, selecting the Button will bring up the Parameters panel for the Button, as seen below.



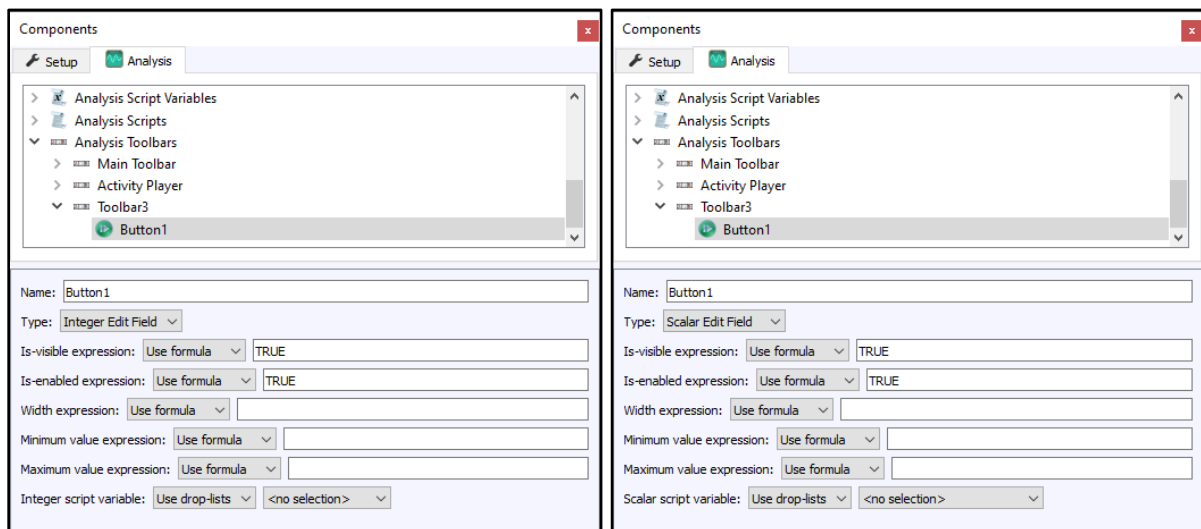
The name of the button will also display when the mouse is positioned over the button in the Toolbar. Boolean expressions for when the Button Is-visible and Is-enabled can also be provided. For instance, TRUE would mean that the button is always visible and enabled. Live would mean that the expression is only TRUE while in the Live window and !Live would make the expression TRUE only within an Activity. Any Boolean expression can be used, thus providing great control over how and when a Button is visible and enabled. Additional Icons can be added by right-clicking on the Button node and selecting the Add Icon option or by clicking on the Add Icon button in the parameters panel when a Button is selected. When using multiple Icons, the Icon index expression will determine which icon is actively displayed for the button based on the specified Integer. The Icon index expression can use an IF Statement to control the resulting Integer value, or it could be an Integer Script variable that obtains its value from the script associated with the Button. The script to be

associated with the Button can be selected from the drop-list. Only scripts from the same tab of the Components window as where the Toolbar resides will be populated in the list.

In addition to being defined as a Button with a script to run, Toolbar Buttons can be configured as a Separator, Playback slider, Label or Integer and Scalar Edit Fields.



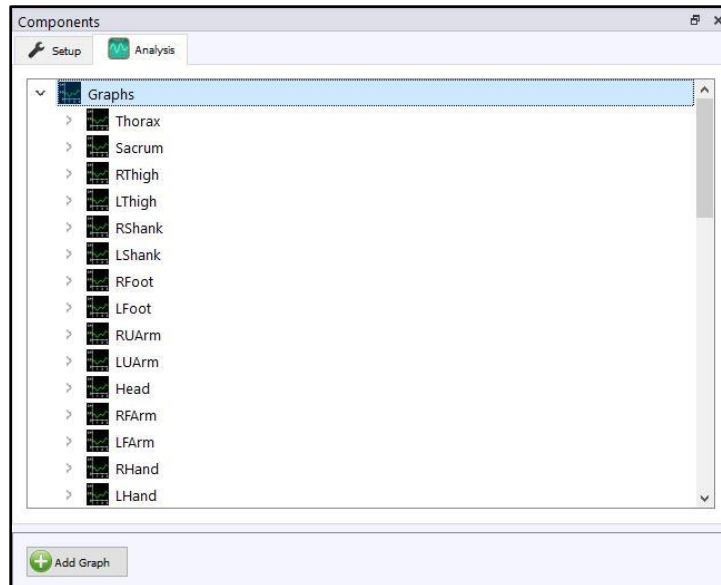
Buttons will be organized within a Toolbar depending how they are listed in the Toolbar node. Buttons can be dragged to new positions within the Toolbar node. The Separator Type will result in a vertical line that separates buttons from before and after the Separator in the Toolbar node. The Playback Slider Type will add a playback slider that moves the time cursor within an Activity. The Label will provide the ability to add text within a Toolbar. Integer and Scalar Edit Fields allow for Integer and Scalar script variables to be updated directly through the Toolbar. For instance, say the speed of an instrumented treadmill is being set by a script variable. The script variable used to control the speed could be updated by entering a value through a Scalar Edit Field in a Toolbar.



For Integer and Script Edit Fields, the Width expression for the edit field displayed in the Toolbar, Minimum and Maximum allowable inputs for the expression and the Integer or Script variable expression can be specified. The script variable needs to reside on the same side of the Components window as the Toolbar.

Next, an example is provided for creating a Toolbar Button that runs a script to hide and show graphs when the Button is clicked. This example would allow the user to easily open and close multiple graphs at once, without having to go into the components window or manually show/hide each graph through the View menu. The steps may easily be adapted and expanded for other graphs or processes.

First, the desired graphs should be created, named, and moved to their desired positions. The graph names displayed in the image below will be used in this example. Revisit the [Adding Graphs Tutorial Video](#) to get started with this process. The video can also be accessed from clicking the Watch Tutorials button on our support page <https://themotionmonitor.com/support/>.



Next, an Analysis Script named ShowHideKinematicsGraphs and a Boolean Analysis Script Variable named IsKinematicGraphsOpen are created. Note that both items are in the Analysis Components tab.

The following script will be used to show and hide the previously mentioned graphs based on their current state. The text below would reside within the ShowHideKinematicsGraphs script. The state of whether the graphs are currently open or closed will be tracked by the Analysis Script Variable IsKinematicGraphsOpen, which will be updated within the script.

```
//Analysis Script Variables used in this script:
//IsKinematicGraphsOpen (Boolean analysis script variable)

// This script controls the display of kinematic graphs. The boolean
IsKinematicGraphsOpen, it should exist in all analyses and toggles the hide and
show below. The script that opens the analysis file containing this script should
set the IsKinematicGraphsOpen boolean to TRUE before making the call. That will
ensure that the analysis has the graphs open when loaded and the toolbar icon
displays a close icon.

//create first condition that if IsKinematicGraphsOpen = TRUE, close the named
graphs

if(IsKinematicGraphsOpen)
{
    //Close Kinematic Graphs
    HideGraph("Head");
    HideGraph("Thorax");
```

```

HideGraph("Sacrum");
HideGraph("RThigh");
HideGraph("LThigh");
HideGraph("RShank");
HideGraph("LShank");
HideGraph("RFoot");
HideGraph("LFoot");
HideGraph("RUArm");
HideGraph("LUArm");
HideGraph("RFarm");
HideGraph("LFarm");
HideGraph("RHand");
HideGraph("LHand");
// Set the condition to false

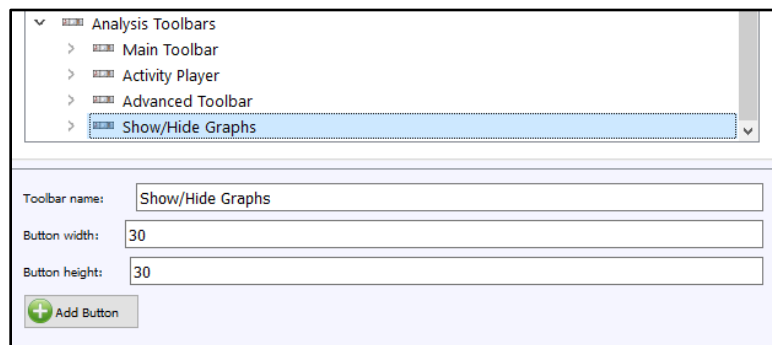
IsKinematicGraphsOpen = FALSE;
}
else
{
//Open desired Kinematic Graphs & (close any other graphs if needed)
ShowGraph("Head");
ShowGraph("Thorax");
ShowGraph("Sacrum");
ShowGraph("RThigh");
ShowGraph("LThigh");
ShowGraph("RShank");
ShowGraph("LShank");
ShowGraph("RFoot");
ShowGraph("LFoot");
ShowGraph("RUArm");
ShowGraph("LUArm");
ShowGraph("RFarm");
ShowGraph("LFarm");
ShowGraph("RHand");
ShowGraph("LHand");

// Set the condition to TRUE
IsKinematicGraphsOpen = TRUE;

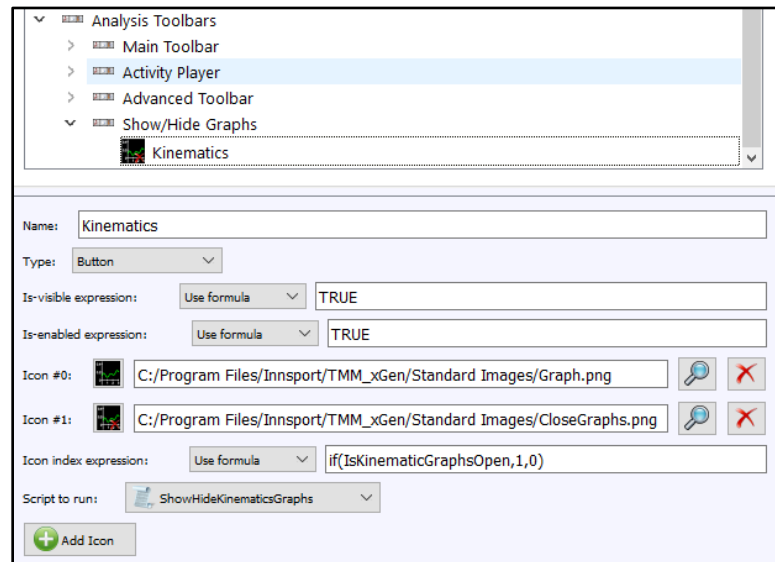
//Exit the else condition
}

```

Next, the Toolbar and Button can be configured. Select Analysis Toolbars from within the Analysis Tab and Click “Add Toolbar”. Since the script and script variable were defined within the Analysis tab of the Components window, the Toolbar will be created here as well. Provide a name for the Toolbar and choose button width and height. Then, click the “Add Button” button.




Below, the Button was named Kinematics, Button was selected for the Type and the Is-visible and Is-enabled expressions were set to always be TRUE.




A second Icon was added, IsKinematicGraphsOpen is used in the Icon index expression to switch between the Icons, and the script to run was selected as ShowHideKinematicsGraphs.

With the parameters panel defined as depicted above, the following will occur when the Button is pressed.

When IsKinematicGraphsOpen is set to TRUE the value for the index is set to 1. The graphs

will open and the icon will display as , which indicates to the user that the button should be pressed again to close the graphs.

When IsKinematicGraphsOpen is set to FALSE the value for the index is set to 0. The

graphs will close, and the icon will display as , which indicates to the user that the button should be pressed again to show the graphs.

Important Note: Graphs can be manually closed by pressing the X in the corner of each graph or manually opened from the View menu. Clicking the button will still act to close or open the graphs, even if the graph is already in that state.