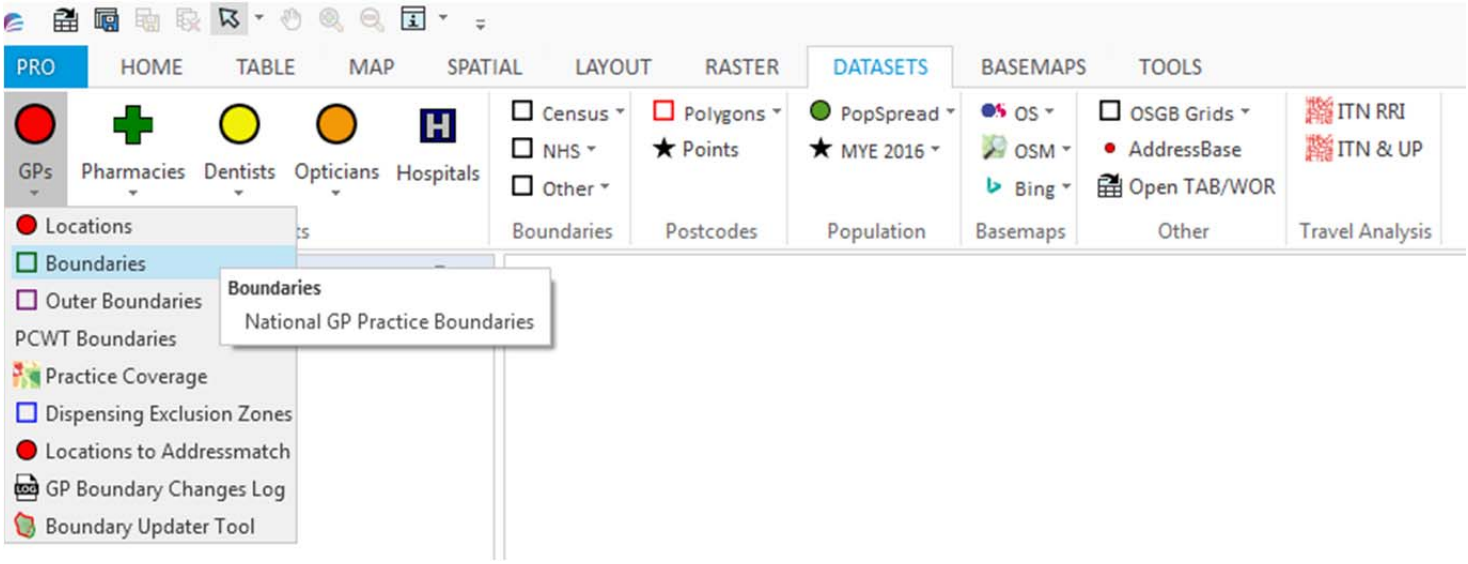
RibbonBuilder for MI Pro 64bit



What is RibbonBuilder?

A tool that builds a custom ribbon interface within MI Pro which can contain any number of datasets, tools or standard MI Pro functions/commands all easily configured via a simple spreadsheet and therefore requires absolutely no MapBasic coding.

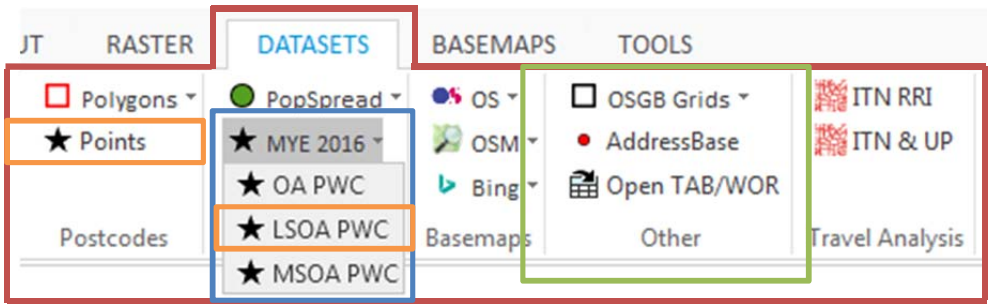
The tool has 2 major functionalities:

1. Allows the administrator to configure a one-stop-shop for all frequently used datasets by the organisation – essentially a 'Favourites' list. This ensures all users in the organisation or team are using the same data sources and don't have to spend time navigating around file systems looking for the correct files.
2. Allows the administrator to configure which tools are automatically loaded on all users MI Pro instances ensuring all users are running the same, up to date versions of all tools. When new MI Pro installations are made the process of adding all the required tools is as simple as adding RibbonBuilder and setting it to AutoLoad.

How does it work?

The administrator configures the required ribbon elements in a spreadsheet adding in tab, group and button names, file paths to .TABs & MBX, links to button icons (external .PNGs or internal MI icons), tooltip info etc. When RibbonBuilder is run for the first time it converts the spreadsheet into a series of .TABs for each of the element types (tab, group, button etc) in the same directory as the MBX. The RibbonBuilder MBX will then loop through the .TAB and build the ribbon to the required configuration. The next time or next user to run RibbonBuilder will read from these pre-generated .TAB files and instantly build the ribbon interface. If changes are required then the spreadsheet can be modified and the **Refresh** function should be run within the RibbonBuilder settings in MI Pro. This will clear the .TABs and rebuild them. (You could also manually clear the .TAB and re-run the RibbonBuilder.MBX)

Ribbon Elements



Key: Tab Group Subgroup Buttons

Note: a splitbutton is not shown as it has the same appearance as a subgroup but has an action assigned to the main button. E.g in the example above, if OSM in the Basemaps group was a split button then clicking it would add a layer to the map but clicking the small down arrow next to it would reveal a dropdown of alternate OSM styles.

How do I add elements to the ribbon?

All elements are configured in an XLSX spreadsheet. A blank template and example spreadsheets are supplied showing the types of elements you can add. The spreadsheet has some basic validation checks built in to help create a valid configuration file. **You should take a copy of either the example or template xlsx, rename to Config.xlsx and then add your content to this file.**

The spreadsheet is split into 5 tabs:

- Tabs
- Groups
- Subgroups
- Buttons
- Tools

These relate to the elements which can be added to the MI Pro ribbon and you should work through them in that order. I.e you must add a custom tab to the 'Tab' tab in Excel first. This new tab name will then appear in the **Groups** tab and allow you to add groups to it. Similarly, you must add a group or subgroup to the correct Excel tab before you can add a button to it and so on.

Within the spreadsheet you will notice some columns and rows are greyed out – this means you should not edit these values or formulas as they will populate automatically. Some columns are green which means there is a drop down list for you to pick a value from (the values within will be defined by the values in other tabs – hence why you should add the elements in the correct oder). The remaining white columns you are free to fill in as required.

Each of the columns in the spreadsheet has a comment attached to the column header which explains what each column does and what values should be entered.

Two things to note are that the "Name" of groups should begin with "G_" and Subgroups begin with "SG_" (see the example config.xlsx). The names of tabs and buttons can be whatever you want, but I'd suggest starting with "T_" and "B_" for clarity.

Subgroup Types

There are two subgroup types:

- **Subgroup** – A subgroup can contain multiple buttons and a dropdown list will appear when the subgroup is clicked.
- **SplitButton** – A splitButton is like a button and subgroup combined. Clicking the splitButton will carry out an action (Add a layer, run a tool etc) but if the small down arrow is clicked then a dropdown list will appear similar to the subgroup

Subgroups and SplitButtons are both added to the **Subgroups** tab of the spreadsheet in the same way. However, if the **ButtonAction** column is left blank then the subgroup will be a regular subgroup. If the **ButtonAction** column is populated with the name of an existing **Button** then it will become a SplitButton and it will inherit the action of the named button.

Button Types

There are two button types:

- **Button** – Just a regular button. It will do the same thing each time you click it. If that is adding a layer then it will try and fail to open and add the layer to the map. If it is running a tool, it will run the tool again.
- **ToggleButton** – This will have 2 actions and only works with the **AddLayer** button action. E.g it will open and add layer to map, or will close and remove from map. Useful for basemaps which you may want to toggle on/off.

Button Actions

Each button can carry out one of 7 actions:

1. **AddLayer** – this will open a .TAB file and add to the current (or new) map. The **Path** column should be populated with the path of the .TAB file.

2. **AddThematicLayer** – this will open a .TAB but not add it to the map. It will instead prompt you to manually add it to the map. This is due to a limitation of Mapinfo/MapBasic in that if a thematic is embedded in the .TAB the style is ignored when added to the map via MapBasic, but is used if manually added. The **Path** column should be populated with the path of the .TAB file.

3. **CustomAction –** This will carry out a specific task that has been coded into the application. There are some CustomActions built into the tool (see example config) but you can also code you own if you are familiar with MapBasic. The **Path** column should be populated with the name of the custom action.

4. **RunTool –** This will launch an external MBX when clicked. The **Path** column should be populated with the path of the .MBX file.

5. **OpenWorkspace –** This will open a workspace. The **Path** column should be populated with the path of the .WOR file.

6. **BrowseTable –** This will open a .TAB file and then browse it. The **Path** column should be populated with the path of the .TAB file.

7. **MenuCommand –** This will run an internal menu command of any of the existing buttons/functions already in MI Pro. The **Path** column should be populated with the number of the MI Pro Menu Command (these can be found in Menu.DEF which will be included in the MapBasic installation directory)

Images and Icons

RibbonBuilder can utilise images and icons from 3 sources:

1. **PNG files** stored in the **Image** folder in the same directory as the MBX. The **Image** column in the spreadsheet should be populated with the filename including extension. E.g GPs.png
   The images should ideally be 32x32 pixels but slightly smaller or larger will also be ok. Note: the same image is used regardless of button size (small or large), in future versions the tool may support 16x16 and 32x32 images to fit each button size.

2. **MI Pro defined icons** – these are the internal icons used by MI Pro. The definitions are found in MI_ICONS_X64.def which is supplied with RibbonLib. E.g MI_IMG_MAP_SQLSELECT_32

3. **Pack URIs** – these are same internal icons as above but referenced by their full pack URI. These can also be found in the MI_ICONS_X64.def file. E.g
   pack://application:,,,/MapInfo.StyleResources;component/Images/Mapping/sqlSelect_32x32.png

FAQ

1. **Why are there 2 MBX for RibbonBuilder? What do they do?**
   The **RibbonBuilder64_Launcher.MBX** is the tool which should be added to all users MI Pro Tools Extension and set to AutoLoad. Once this is added, the user shouldn't need to ever change this for any future updates or refreshes of this or any other linked tools.
   RibbonBuilder64_Launcher.MBX will run on a user's machine and load the most recent version of RibbonBuilder MBX e.g (**RibbonBuilder64_v0.52.MBX**). The version which will run is set by the administrator within the source code of the launcher MBX. Since the launcher runs and quits, the launcher MBX is never locked which means the administrator is free to compile new versions of the tool without having to ask all users to close the tool to unlock the file or update the tool path in their MI Pro Tools Extension.

2. **What is the difference between the Reload and Refresh buttons on the HOME tab?**
   The **Reload** button will just reload the RibbonBuilder tool and re-add all the elements to the MI Pro ribbon. This will usually be done if something goes wrong, e.g a tool crashes and a button disappears. The **Refresh** button will refresh the configuration files and reload RibbonBuilder. This should be used when you have added/modified content in the Config.xlsx and want to see the changes in MI Pro.

3. **What is the difference between adding a tool to the Tools tab and adding it to the Buttons tab in Config.xlsx?**

Adding an MBX as a button will only run the MBX when the user clicks the button on the ribbon. This should be used for tools that **don't** add their own buttons to the interface e.g simple tools that run immediately and carry out a specific task.

MBX tools that add their own ribbon elements should be added to the **Tools** tab of the config. This means RibbonBuilder will run the MBX when it first loads which will load your custom tool and add any ribbon elements it may have. This is best used for more complex tools which you usually have set to AutoLoad in MI Pro.

4. **Why would I use the Tools tab in Config.xlsx when I already have the tools registered and set to AutoLoad in the MI Pro Tool Extension?**
   You can of course leave your tools as they are and not use the Tools tab in the config.xlsx. The purpose of this feature is to make it easier to centrally administrate the tools/versions used by all MI Pro users. Take for example a team of 30 MI Pro users using 15 tools. Every time a new version of a tool is released everyone has to manually change the filepath to the tool. If left to the users some may not update it and it becomes a real chore for an administrator to go around and manually update all users. Using this RibbonBuilder feature the administrator can define all the tools which should be loaded on all the MI Pro instances and then easily update paths in 1 place when they change. This ensures all users are always up to date and not using different/outdated versions of tools. It also makes it much easier to rollout new MI Pro installs as you can just load RibbonBuilder and it will automatically load all your organisations standard tools.

5. **Can I prevent all users from refreshing/editing the config**?
   Yes, remove the **Refresh** and **Edit** buttons as well as the **Config** subgroup from **Config.XLSX.**
   If an administrator wishes to refresh the config then they should delete the Tab.tab, Groups.tab, Subgroups.tab, Buttons.tab and Tools.tab from the application directory and then run RibbonBuilder which will rebuild the .tab files based on the current config. Any user could of course do the same, so as an extra precaution you should also store the config file in a restricted location as well as ensure the .tab files the tool creates (Tabs.tab, Groups.tab etc) are set to read only.

6. **Can I have different configurations for different users/teams?**
   Only 1 configuration can be used at a time with the MBX but you can keep multiple config spreadsheets which you can swap in/out and then **Refresh** the tool from the HOME tab.

   If you need to run multiple configurations for different users/teams the solution is to create a copy of the 2 MBX files and the Config.xlsx and save in a new location. Then configure the spreadsheet to suit a particular user/team. Then each user/team should load their relevant version of RibbonBuilder.

7. **What does the LayerScale value do in the Config.XSLX button tab?**
   We've all accidentally opened a huge dataset and it automatically gets added to the current map which is currently set at a national scale and then Mapinfo locks up and then crashes taking all your unsaved progress with it. This feature attempts to avoid that situation by warning you that you're about to load a dataset at the wrong scale. The value should be set to a realistic viewing scale for this particular dataset. E.g OS 50k Raster should be set to 50000. A warning will be

triggered if a user attempts to open this layer at more than **2x** the scale, in this case at a scale over 1:100,000. If no map is open when the user opens a layer, the **LayerScale** value will be used as the scale of the new map when it opens.

8. **Can I add custom functionality to the tool? E.g a button that launches an external script, or opens a certain file?**
   Yes, the source code is provided so you can write your own custom functions in MapBasic. You can see examples of custom functionality in the **runCustomCommand** Sub in the source code. If it's a simple task then you can add it here but for more complex tasks it's probably better to create a separate MBX and add it as a button to config.xlsx

9. **Can I add elements to the default MI Pro tabs/groups/subgroups?**
   Yes, as long as they are listed in the config spreadsheet you can add to the existing ribbon elements. For example, on the 'Tabs' tab of the spreadsheet you should see the standard MI Pro tabs are already listed meaning you can add buttons etc to these tabs. However, if you wish to add buttons to a standard group or subgroup then they should be added to relevant tab in the spreadsheet with the correct MI Pro internal name before you can add elements to them. (These internal names are listed in RibbonElements.def within RibbonLib)

10. **Can I add the same button to more than one location?**
    Yes, see the example config file where all the basemap buttons/layers appear on their own **BASEMAPS** tab, but also in a dropdown menu on the **DATASETS** tab. To do this, just duplicate the row for the button in the spreadsheet and change the options so it appears in a different location. Note: the **DisplayName** value can be the same for both buttons, but the **Name** value should be unique to avoid conflicts.

Troubleshooting

- **The tool fails to even start**
  Do you have write access to the directory where **RibbonBuilder64_Launcher.MBX** is stored? The tool needs to create some .TAB files in this location so will fail if you have no write access.

- **The tool starts to load, tabs/buttons start to appear but then they all disappear.**
  This suggests that one of the tools that RibbonBuilder was trying to load isn't valid or won't open which causes RibbonBuilder to quit. Check all filepaths in the **Tools** tab of the **spreadsheet** are valid and the MBX work when opened manually.

- **My icons seem to be wrong even though they are correct in the spreadsheet.**
  In the spreadsheet, check the button/group/subgroup in the next row **after** the button/group/subgroup with the incorrect icon. This is because if the next row in the spreadsheet is invalid and the element isn't added correctly then the attributes of the next button can sometimes be applied to the last successfully added button which causes this mismatch.

Compiling from source code

RibbonBuilder relies on **RibbonLib** which is part of Peter Horsbøll Møller's **mbLibrary** which can be found here: https://github.com/PeterHorsbollMoller/mbLibrary

You should download and extract mbLibrary and then modify the source code of RibbonBuilder to point to the correct location of the required libraries on your machine.

You will need to update the paths in:

- RibbonBuilder.mbp
- RibbonBuilder.mb