# Graphical User Interface

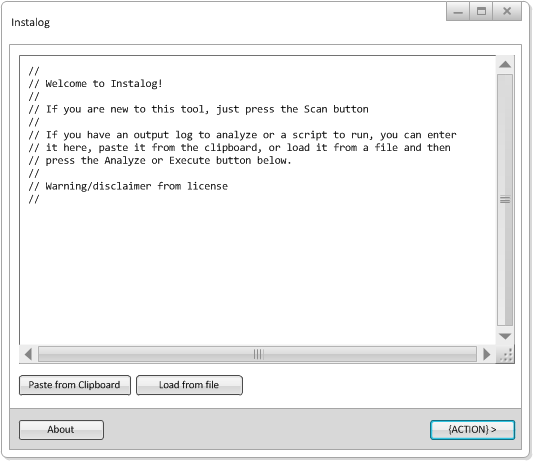
## Background

The Graphical User Interface (GUI) shall be designed in such a way that it can accommodate the various usage scenarios common for this tool. Therefore, it must bridge the gap between simplicity and complexity so that users can simply use the tool and power users can use the tool to create powerful system-altering scripts.

The GUI is inspired by the well-known Windows application installer paradigm. Basic users will only encounter screens similar to what they are familiar with when installing or uninstalling applications. The interface will become much more complex when power users use the tool to modify scripts, but this is to be expected. The editing script interface is inspired by tools that exist in the field (namely OTA and HijackThis).

## Main Screen

When a user opens the application, they shall be presented with the following screen:



### Textbox requirements

1. The textbox should support basic operations including but not limited to Copy, Paste, and Undo/Redo.
2. The textbox should not implement word-wrap.

### Paste from Clipboard button requirements

1. This button should clear the contents of the textbox and then place the full contents of the clipboard into the textbox
2. If the clipboard does not contain text data, this button should be disabled

### Load from file button requirements

1. Pressing this button will the standard Windows file open dialog. It should be enabled for searching for \*.txt and \*.zip files.
   1. If a valid file is opened, the contents of the textbox should be cleared and then the full contents of the file should be placed into the textbox. Obviously, zipped files should be unzipped.
   2. If the user presses cancel in the dialog, the contents of the textbox should not be changed

### {ACTION} button requirements

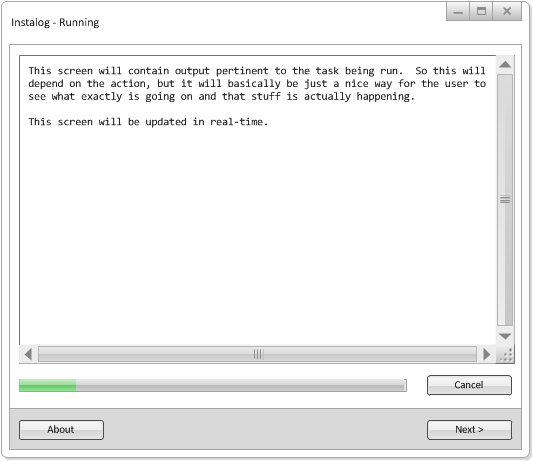
1. The contents of the textbox should be scanned to determine if it contains default script content (or empty script content), script content, or log content. If so, the button should display “Scan,” “Execute,” or “Analyze” (respectively).
2. The button should have differing behavior based on the inferred content of the textbox:
   1. If the button reads “Scan,” the tool should proceed to the [Running Screen](#_Running_Screen) running the default script.
   2. If the button reads “Execute,” the tool should proceed to the [Running Screen](#_Running_Screen) running the supplied script.
   3. If the button reads “Analyze,” the tool should proceed to the [Analysis Screen](#_Analysis_Screen) displaying the parsed log.
3. If the type of the content in the textbox cannot be inferred or the content of the textbox is not syntactically valid, the button should be disabled and should display the text of the last inferred {ACTION}.
4. If the content of the textbox is a script that is not for the user’s computer, the {ACTION} button should show “Execute,” but it should be disabled. Hovering over the {ACTION} button should show a tooltip explaining why.

### About button requirements

1. This button should bring up a screen that contains any licensing information from the license for this project as well as information for any other tools used in this project. This screen should have a simple close button.
2. The behavior for this button is the same on all following windows.

## Running Screen

This screen will run a script. For the purpose of these requirements, it is not important whether the script is the default script or a custom script. The script will automatically begin when this screen is displayed. The Running Screen is presented below:



### Textbox requirements

1. The textbox should be updated to display the (raw) log output of the currently running screen. This textbox should only be updated at a rate of 60 Hz to avoid slowing down the GUI.
2. The textbox should be scrollable. It should automatically scroll down to follow the output by default. If the user scrolls up for any reason, it should no longer auto-scroll unless the user manually scrolls all the way down to the bottom of the output.

### Progress bar requirements

1. The progress bar should contain the best estimate of the progress through the script. This estimate can be something as simple as the completed script actions divided by the total script actions.

### Cancel button requirements

1. The “Cancel” button should not be enabled if the script contains any system-altering actions. This should be determined by scanning the script in advance.
2. If the user presses the “Cancel” button, the “Next” button should change to display “Exit.” The text of the “Cancel” button should change to “Re-run.” The behavior of both of these buttons should be self-explanatory.
3. When the script completes, if the script was not a system-altering script, then the button should change to “Re-run.” Otherwise, it should remain displaying “Cancel” and should become disabled.

### Window Close Button Requirements

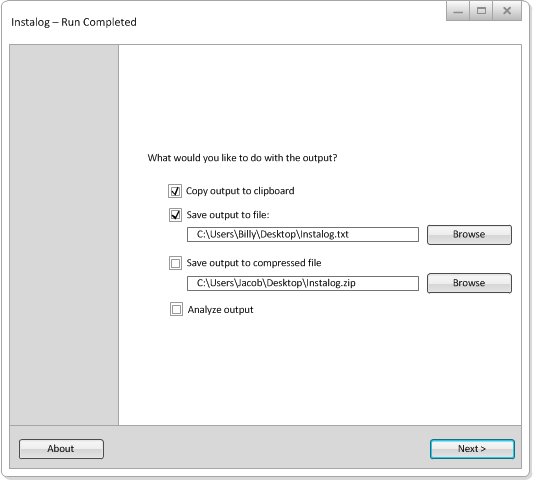
1. The window close button should not be enabled if the script contains any system-altering actions. This should be determined by scanning the script in advance.
2. If the user presses the window close button at any time that it is enabled, a Yes/No dialog should appear that reminds the user that the script output has not been saved yet.

### Next button requirements

1. The “Next” button should not be enabled until after the script completes.
2. When the user presses the “Next” button, the tool should proceed to the [Run Completed Screen](#_Run_Completed_Screen).

## Run Completed Screen

This screen enables a user to decide what to do with their script output (log file).



### Option Requirements

1. By default, nothing should be selected
2. Both of the file Save fields should default to the path that the Instalog executable is run from. For example, if the executable were being run from C:\Users\Billy\Desktop, the Save fields should default to C:\Users\Billy\Desktop\Log.txt and C:\Users\Billy\Desktop\Log.zip.

### Next Button Requirements

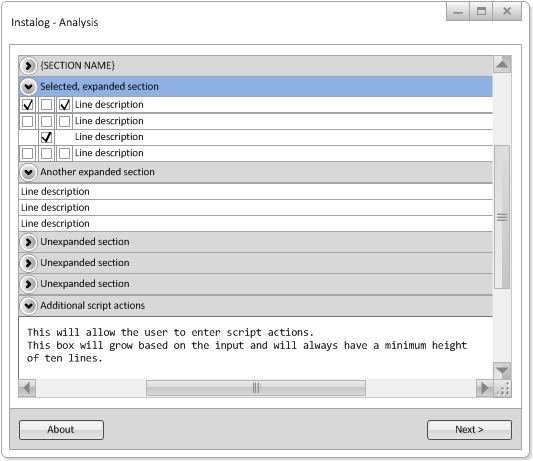
1. The button should be disabled until at least one option is selected. It should return to being disabled if nothing is selected.
2. The button should have the following behavior when it is clicked:
   1. If the options selected did not include “Analyze output,” then the tool should proceed to the [Finished Screen](#_Finished_Screen).
   2. If the options selected include “Analyze output” and other options, then the other options should execute and then the tool should proceed to the [Analysis Screen](#_Analysis_Screen).
   3. If the only option selected was “Analyze output,” then the tool should proceed to the [Analysis Screen](#_Analysis_Screen).

### Window Close Button Requirements

1. If the user presses the window close button at any time that it is enabled, a Yes/No dialog should appear that reminds the user that the script output has not been saved yet.

## Analysis Screen

The analysis screen enables users to construct a script based on the output from a log. For the requirements listed in this section, it does not matter what workflow the user used to get to this screen.



### Section Requirements

1. Each separate section in the log will be parsed into one of the gray section. Sections in the log are denoted by ================== {SECTION NAME} ==================.
2. Sections can be collapsed or expanded by the user. The user can either press the entire section heading or the Chevron arrow to perform this action. By default, all sections will be expanded.
3. The Chevron arrow should point to the right for collapsed sections and downward for expanded sections.
4. No animation is necessary for the collapse action or expand action.
5. Hovering over a section should slightly change the background color.
6. Each corresponding line should be listed under the section. Some sections might not have any lines. In this case, a single line should appear with the following text in italics: “No lines available for this section.”

### Line Requirements

1. Each separate line of the log will be logged into a line underneath the corresponding section.
2. Hovering over a line should slightly change the background color.

### Checkbox Requirements

1. Depending on the line, there may be zero, one, or many actions available for the given line. There should be a checkbox to the left of the line for each action. Checking a box indicates that the action should be taken.
2. A user should be able to hover their mouse over any checkbox to determine what action the checkbox enables.
3. In a given section, actions should be grouped by actions. Therefore, each column will only have one type of action in it. If an action does not apply to a line, there should simply be an empty slot where the checkbox would be.
4. All checkboxes should default to unchecked.

### Additional script actions section

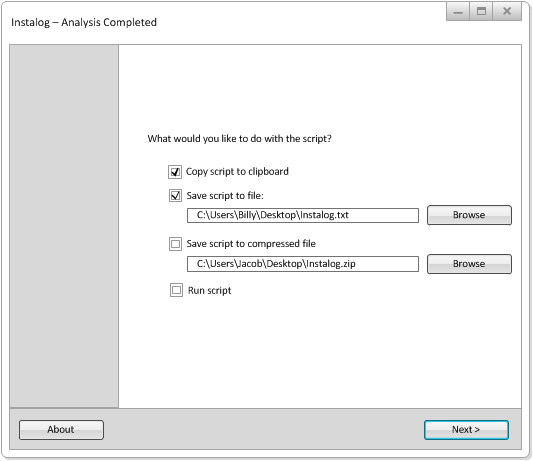
1. The last section in the log will always be titled “Additional script actions” and will contain a textbox that will allow the user to specify additional script actions to take
2. The textbox should always have a minimum height of ten lines and will grow to always be one line longer than its contents
3. The textbox should not have its own scrollbars. Rather, the scrollbars for the rest of the control should be sufficient
4. The textbox should support basic operations including but not limited to Copy, Paste, and Undo/Redo.
5. The textbox should not implement word-wrap.

### Next button requirements

1. The Next button should display a Yes/No dialog warning the user that scripts are final and there is no going back.

## Analysis Completed Screen

This screen enables a user to decide what to do with the finished script.



### Option Requirements

1. By default, nothing should be selected
2. Both of the file Save fields should default to the path that the Instalog executable is run from. For example, if the executable were being run from C:\Users\Billy\Desktop, the Save fields should default to C:\Users\Billy\Desktop\Script.txt and C:\Users\Billy\Desktop\Script.zip.
3. The “Run script” option should be disabled if the log the script is based on is not from the computer that built the script. Hovering over the option should show a tooltip explaining why.

### Next Button Requirements

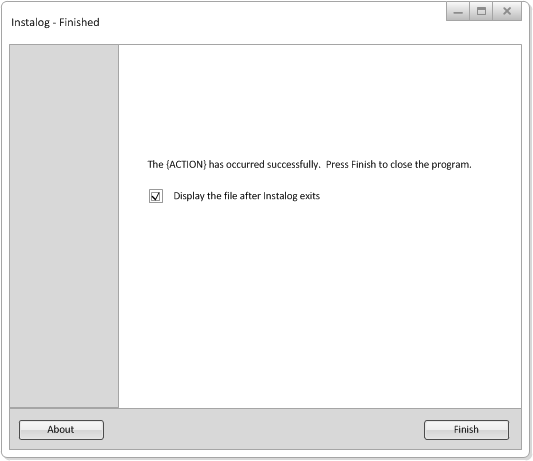
1. The button should be disabled until at least one option is selected. It should return to being disabled if nothing is selected.
2. The button should have the following behavior when it is clicked:
   1. If the options selected did not include “Run script,” then the tool should proceed to the [Finished Screen](#_Finished_Screen).
   2. If the options selected include “Run script” and other options, then the other options should execute and then the tool should proceed to the [Running Screen](#_Running_Screen).
   3. If the only option selected was “Run script,” then the tool should proceed to the [Running Screen](#_Running_Screen).

### Window Close Button Requirements

1. If the user presses the window close button at any time that it is enabled, a Yes/No dialog should appear that reminds the user that the script has not been saved yet.

## Finished Screen

The finished screen provides the user with some confirmation that the actions instructed of the tool were actually executed. This is to prevent users from becoming disoriented and thinking that the tool had crashed or something else bad had happened.



### Text Requirements

1. The text should display some information about what happened. The text should cover all possible combinations of copying material to the clipboard, saving one file, or saving two files.

### Display File Checkbox Requirements

1. If a plaintext file was saved, then a checkbox should be displayed that allows the user to specify if they want to view the file when the program exits.
2. The default state of this checkbox should be unchecked.
3. Upon exiting, if this is checked, notepad should be launched and display this file.

## GUI Flowchart Diagram

Since there are several different branch points in the tool, the flow through the tool is difficult to describe by simply using text. The flow described in the preceding sections is described below in flowchart form:

