March 2020

Automatically Generating Text-based Commands From Actions Performed Via GUI

Stefan Lindmark

Automatically Generating Text-based Commands From Actions Performed Via GUI

ABSTRACT

Graphical user interfaces (GUIs) are inefficient and cumbersome for tasks that need to be performed repeatedly and/or on a large scale. In such situations, a user can benefit from automating sequences of actions in the form of text-based commands or scripts. Translating familiar GUI operations into equivalent text commands and/or API function calls often requires consulting documentation from a separate information source. Identifying the appropriate command and its task-specific parameters within such documentation is tedious and time-consuming. This disclosure describes functionality to assist users in converting actions performed via a GUI into corresponding text commands and/or API calls. With user permission, each user action within the GUI is translated to an equivalent command. The dynamically updated sequence of such commands can be presented in a separate section within the user interface.

KEYWORDS

- Automation
- Virtual assistant
- Assistive technology
- Macro
- Shell script
- Batch processing
- Action sequence

BACKGROUND

Many systems and applications permit performance of actions via both an interactive graphical user interface (GUI) and a non-interactive application programming interface (API) that make the functionality available via programs, scripts, text-based commands, etc. GUIs serve as an effective mechanism for enacting actions and promoting understanding of system operation by end users, especially novices. However, GUIs are inefficient and cumbersome for tasks that need to be performed repeatedly and/or on a large scale. For instance, system administrators often need to perform the same set of operations for each user of the systems they manage. In such cases, an administrator can benefit from automating actions or sequences of actions in the form of scripts that can repeat the action(s) for multiple users and/or objects (e.g., files). Moreover, the administrator can implement the script to make dynamic decisions based on the state of the system at runtime.

Creating such scripts requires access system functionality via text commands and/or API calls equivalent to the operation performed via the GUI. Translating familiar GUI operations into equivalent text commands and/or API function calls requires acquiring additional knowledge by consulting documentation from a separate information source, such as API documentation. Identifying the appropriate command within such documentation is often tedious and time-consuming because it requires guessing, searching, browsing, etc. After finding a suitable command, the user needs to further read description of all variants and options pertaining to the operation in order to determine the parameters needed to use the command for the desired task.

In some cases, a manual action performed via the GUI is not available via a text command or API. In such cases, the lack of a non-GUI option for the desired operation can be

confirmed only by an exhaustive search of the API documentation, unless the documentation includes a page explicitly mentioning that specific functionality is available only within the GUI.

The challenges of locating desired information within API documentation can often lead users to seek information from unofficial alternate sources, such as online user forums. Information obtained via such sources can be outdated and/or inaccurate. For instance, example code provided by others in response to a user's query posted to a forum can contain bugs or have other problems.

Many operating systems and applications provide keyboard shortcuts for common actions or sequences of actions. Alternatively, or in addition, many operating systems and applications include functionality to record GUI actions in a form that can be bundled and executed to facilitate batch processing and automation. Examples of such functionality include mouse movement and click recording, macros generated from GUI action sequences, processes created via a GUI interface to specify operation sequences, etc. The functionality can further permit users to assign keyboard shortcuts to such user-created action sequences.

Further, many systems include wizards that provide a structured approach for executing a sequence of actions based on parameters provided to the wizard. However, users may not be aware of the existence of wizards related to their desired tasks. Moreover, the code for the wizard may be separate from that for the GUI, thus making it difficult for the wizard and the GUI to stay in sync. For instance, the user often needs to provide extensive input to the wizard as it typically does not pre-populate parameters based on the content within the application's main GUI that is relevant to the user's task.

DESCRIPTION

This disclosure describes functionality to assist users in converting actions performed via a GUI into corresponding text commands and/or API calls. With user permission, user interactions with the GUI are evaluated and translated to corresponding equivalent text-based command and/or API calls. The sequence of such commands, generated with user permission, can then be presented in a separate section within the application interface.

If the user permits, the contents in the separate section are updated dynamically and continually as the user goes about performing tasks via the GUI. The application GUI can be further augmented to include icons that signal the availability of text-based commands connected to actions available within the GUI. Such icons can serve to facilitate the discovery of alternate execution options and promote the use of automation for increased efficiency.

To avoid cluttering the GUI, the section containing the dynamically generated command sequences can be collapsed into an icon within the interface. When a user encounters a task that can benefit from automation, the user can interact with the collapsed icon to expand the command section and reveal the text-based commands and/or API calls corresponding to the GUI actions for the task. The text-based commands can be provided in a number of equivalent forms covering a variety of purposes, contexts, languages, etc. For instance, the commands can be presented as various alternative versions, such as REST API calls, shell commands, script code, etc. Users can select the form that is suitable for their needs and execute the respective command, e.g., by copying to the command line and/or copying to a script.

If the user permits, the above described mechanism can also infer the user's intended tasks based on the sequence of GUI actions. Captured actions sequences that include tasks with potential for automation can be flagged and brought to the user's attention via suitable

mechanisms, e.g., menu bar of the application. A user that inspects the flagged information is shown a historical sequence of text commands equivalent to the manual actions performed within the GUI. The user can then select and copy the appropriate set of commands needed to automate the higher level task at hand.

For instance, making several types of changes to a device typically entails multiple interactions within the GUI. Availability of the historical sequence of text commands enables the user to copy the entire set of commands needed to make all of the needed changes instead of needing to copy each command separately after enacting each individual change via the GUI.

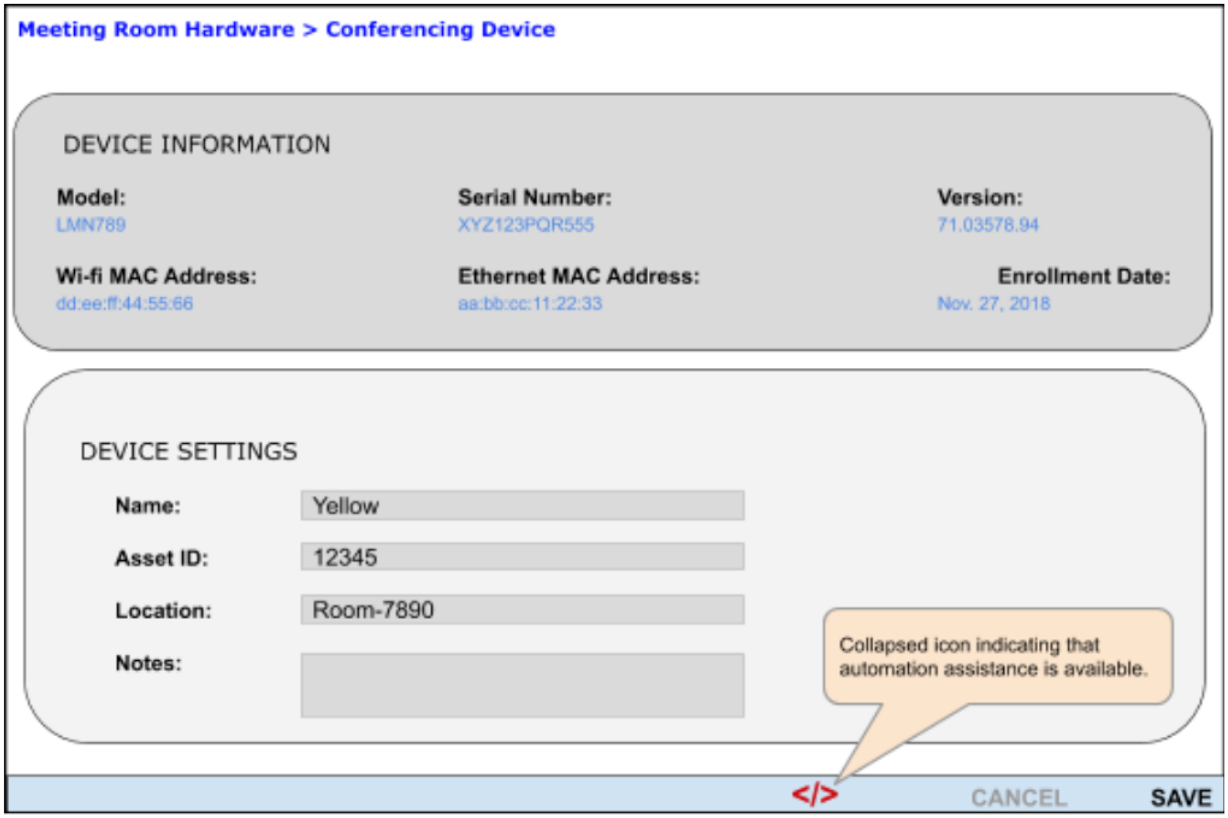

Fig. 1: Assistive functionality for automation accessible within an application

Fig. 1 shows an example of an application for administering meeting room hardware within an enterprise facility. The application is augmented with the assistive functionalities

described in this disclosure. Assistance for translating GUI actions to text based commands is accessible via a collapsible icon, shown in red in Fig. 1.

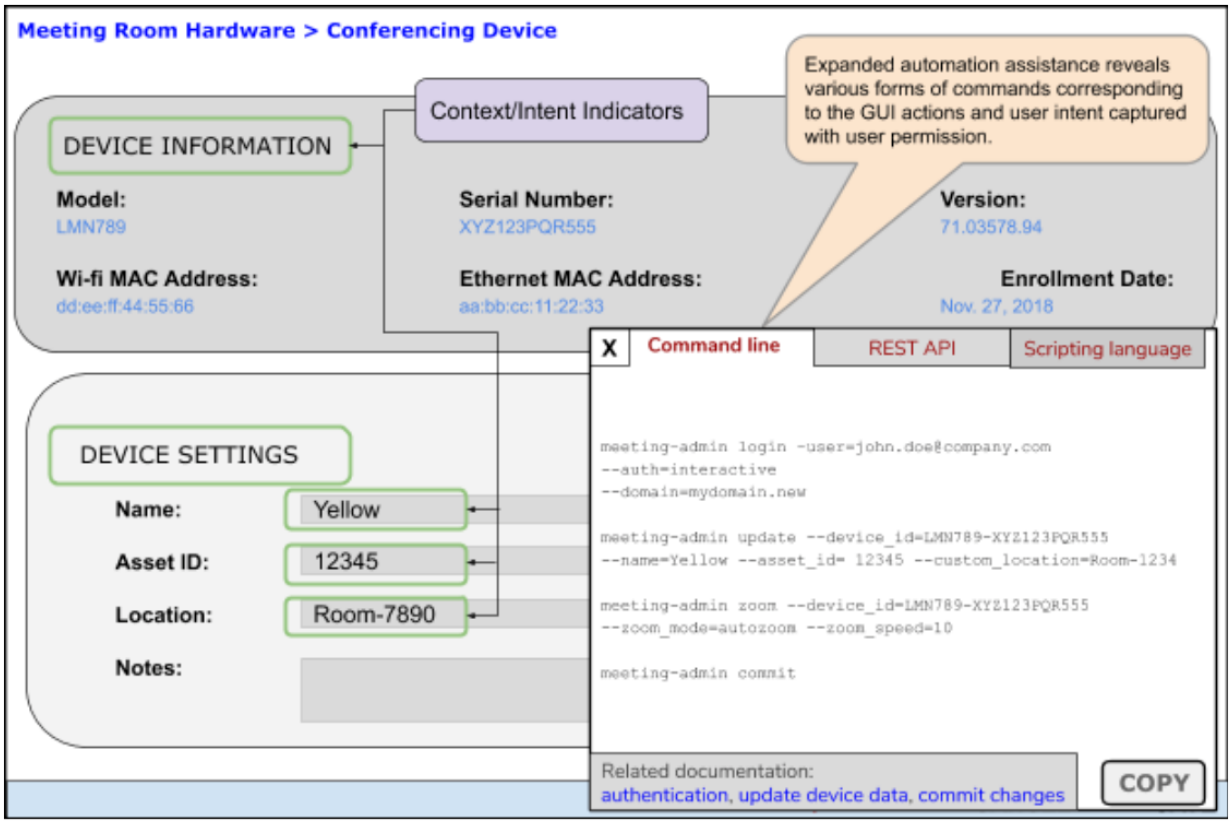

Fig. 2: Sequence of text-based commands equivalent to the most recent GUI actions

Fig. 2 illustrates operational details of invoking the assistive functionality by clicking collapsible icon to reveal a sequence of text-based commands corresponding to the history of user actions within the GUI captured with the user's permission. Specifically, Fig. 2 illustrates the task of renaming a number of hardware devices in meeting rooms within the enterprise facilities. The user first employs the GUI to change the name of the first device within a long list of devices whose names are to be updated.

Upon determining that the same set of actions is to be repeated for each device in the list, the user seeks automation assistance by clicking the collapsed icon to inspect the text commands

corresponding to the recent device name change actions performed via the GUI. The commands are available in one or more forms such as command line, REST API, and scripting language. If the user permits, variable parameters for the commands or function calls are filled in based on permitted user data and the inferred intended task. For instance, the device ID can be taken from the currently loaded device information page and the new name for the device can be set to the text entered manually within a corresponding text box presented by the GUI.

The assistive techniques described in this disclosure offer a number of advantages, such as:

- promoting awareness of the possibilities for automating actions taken within the GUI,
- providing validated and accurate commands and/or code needed to execute the desired tasks,
- minimizing the need to consult documentation and online help, thus saving time and effort, and
- improving the user experience of the application by integrating automation related functionality within the GUI, thus reducing switches between the application, documentation, and the command line.

The advantages can lead users to engage in higher levels of automation. In an enterprise context, the corresponding reduction in time and effort across users can add up to resource savings and increased operational efficiency.

Further to the descriptions above, a user is provided with controls allowing the user to make an election as to both if and when systems, programs or features described herein may enable collection of user information (e.g., information about a user's interaction with an application GUI). In addition, certain data may be treated in one or more ways before it is stored

or used, so that personally identifiable information is removed. For example, a user's identity may be treated so that no personally identifiable information can be determined for the user. Thus, the user has control over what information is collected about the user, how that information is used, and what information is provided to the user.

CONCLUSION

This disclosure describes functionality to assist users in converting actions performed via a GUI into corresponding text commands and/or API calls. With user permission, each user action within the GUI is translated to an equivalent command. The dynamically updated sequence of such commands can be presented in a separate section within the user interface, e.g., in a separate section that can be accessed via a collapsible icon. The application GUI can be further augmented to include icons that signal the availability of text-based commands connected to actions available within the GUI. The resulting improved user experience can lead users to engage in higher levels of automation. In an enterprise context, the corresponding reduction in time and effort across users can provide resource savings and increased operational efficiency.

REFERENCES

1. Harm, Michael W., and Micah Lemonik. "Server side macro method, apparatus and computer readable medium." U.S. Patent Application 14/255,718, filed October 23, 2014.