UX checklist for desktop applications

05/31/2018 • 24 minutes to read •

In this article

- Windows
- Layout
- Text
- Controls
- Keyboard
- Mouse
- Dialog boxes
- Property Sheets
- Wizards
- Wizard pages
- Error messages
- Warning messages
- Confirmations
- Icons
- Help

Here's a collection of some of the important guidelines in UX Guide. You can use this as a checklist to make sure your program user interface gets these important items right.

Windows

- **Support the minimum Windows effective resolution of 800x600 pixels.** For critical user interfaces (UIs) that must work in safe mode, support an effective resolution of 640x480 pixels. Be sure to account for the space used by the taskbar by reserving 48 vertical relative pixels for windows displayed with the taskbar.
- **Optimize resizable window layouts for an effective resolution of 1024x768 pixels.** Automatically resize these windows for lower screen resolutions in a way that is still functional.
- **Be sure to test your windows in 96 dots per inch (dpi) (at 800x600 pixels), 120 dpi (at 1024x768 pixels), and 144 dpi (at 1200x900 pixels) modes.** Check for layout problems, such as clipping of controls, text, and windows, and stretching of icons and bitmaps.
- **For programs with touch and mobile use scenarios, optimize for 120 dpi.** High-dpi screens are currently prevalent on touch and mobile PCs.

- **If a window is an owned window, initially display it "centered" on top of the owner window.** Never underneath. For subsequent display, consider displaying it in its last location (relative to the owner window) if doing so is likely to be more convenient.
- **If a window is contextual, always display it near the object that it was launched from.** However, place it out of the way (preferably offset down and to the right) so that the object isn't covered by the window.

Layout

- **Size controls and panes within a window to match their typical content.** Avoid truncated text and their associated ellipses. Users should never have to interact with a window to view its typical content—reserve resizing and scrolling for unusually large content. Specifically check:
  - **Control sizes.** Size controls to their typical content, making controls wider, taller, or multi-line if necessary. Size controls to eliminate or reduce scrolling in windows that have plenty of available space. Also, there should never be truncated labels, or truncated text within windows that have plenty of available space. However, to make text easier to read, consider limiting line widths to 65 characters.
  - **Column widths.** Make sure list view columns have suitable default, minimum, and maximum sizing. Choose list views default column widths that don't result in truncated text, especially if there is space available within the list view.
  - **Layout balance.** The layout of a window should feel roughly balanced. If the layout feels left-heavy, consider making controls wider and moving some controls more to the right.
  - **Layout resize.** When a window is resizable and data is truncated, make sure larger window sizes show more data. When data is truncated, users expect resizing windows to show more information.
- **Set a minimum window size if there is a size below which the content is no longer usable.** For resizable controls, set minimum resizable element sizes to their smallest functional sizes, such as minimum functional column widths in list views.

Text

- **Use ordinary, conversational terms when you can.** Focus on the user goals, not technology. This is especially effective if you are explaining a complex technical concept or action. Imagine yourself looking over the user's shoulder and explaining how to accomplish the task.
- **Be polite, supportive, and encouraging.** The user should never feel condescended to, blamed, or intimidated.

- **Remove redundant text.** Look for redundant text in window titles, main instructions, supplemental instructions, content areas, command links, and commit buttons. Generally, leave full text in main instructions and interactive controls, and remove any redundancy from the other places.
- **Use title-style capitalization for titles, and sentence-style capitalization for all other UI elements.** Doing so is more appropriate for the Windows tone.
  - **Exception:** For legacy applications, you may use title-style capitalization for command buttons, menus, and column headings if necessary to avoid mixing capitalization styles.
- **For feature and technology names, be conservative in capitalizing.** Typically, only major components should be capitalized (using title-style capitalization).
- **For feature and technology names, be consistent in capitalizing.** If the name appears more than once on a UI screen, it should always appear the same way. Likewise, across all UI screens in the program, the name should be consistently presented.
- **Don't capitalize the names of generic user interface elements,** such as toolbar, menu, scroll bar, button, and icon.
  - **Exceptions:** Address bar, Links bar, ribbon.
- **Don't use all capital letters for keyboard keys.** Instead, follow the capitalization used by standard keyboards, or lowercase if the key is not labeled on the keyboard.
- **Ellipses mean incompleteness.** Use ellipses in UI text as follows:
  - **Commands.** Indicate that a command needs additional information. Don't use an ellipsis whenever an action displays another window—only when additional information is required. Commands whose implicit verb is to show another window don't take an ellipsis, such as Advanced, Help, Options, Properties, or Settings.
  - **Data.** Indicate that text is truncated.
  - **Labels.** Indicate that a task is in progress (for example, "Searching...").

Tip: Truncated text in a window or page with unused space indicates poor layout or a default window size that is too small. Strive for layouts and default window sizes that eliminate or reduce the amount of truncated text. For more information, see Layout.

- **Don't use blue text that isn't a link, because users may assume that it is a link.** Use bold or a shade of gray where you'd otherwise use colored text.
- **Use bold sparingly to draw attention to text users must read.**
- **Use the main instruction to explain concisely what users should do in a given window or page.** Good main instructions communicate the user's objective rather than focusing just on manipulating the UI.
- **Express the main instruction in the form of an imperative direction or specific question.**

- **Don't place periods at the end of control labels or main instructions.**
- **Use one space between sentences.** Not two.

Controls

- **General**
  - **Label every control or group of controls. Exceptions:**
    - Text boxes and drop-down lists can be labeled using prompts.
    - Subordinate controls use the label of their associated control. Spin controls are always subordinate controls.
  - **For all controls, select the safest (to prevent loss of data or system access), most secure value by default.** If safety and security aren't factors, select the most likely or convenient value.
  - **Prefer constrained controls.** Use constrained controls like lists and sliders whenever possible, instead of unconstrained controls like text boxes, to reduce the need for text input.
  - **Reconsider disabled controls.** Disabled controls can be hard to use because users literally have to deduce why they are disabled. Disable a control when users expect it to apply and they can easily deduce why the control is disabled. Remove the control when there is no way for users to enable it or they don't expect it to apply, or leave it enabled, but provide an error message when it is used incorrectly.
    - **Tip:** If you aren't sure whether you should disable a control or provide an error message, start by composing the error message that you might provide. If the error message contains helpful information that target users aren't likely to quickly deduce, leave the control enabled and provide the error. Otherwise, disable the control.
- **Command buttons**
  - **Prefer specific labels over generic ones.** Ideally users shouldn't have to read anything else to understand the label. Users are far more likely to read command button labels than static text.
    - **Exception:** Don't rename the Cancel button if the meaning of cancel is unambiguous. Users shouldn't have to read all the buttons to determine which button cancels an action. However, rename Cancel if it is unclear what action is being canceled, such as when there are several pending actions.
  - **When asking a question, use labels that match the question.** For example, provide Yes and No responses to a yes or no question.
  - **Don't use Apply buttons in dialog boxes that aren't property sheets or control panel items.** The Apply button means apply the pending changes, but leave the window open. Doing so allows users to evaluate the changes before

closing the window. However, only property sheets and control panel items have this need.
    - Label a button Cancel if canceling returns the environment to its previous state (leaving no side effect); otherwise, label the button Close (if the operation is complete), or Stop (if the operation is in progress) to indicate that it leaves the current changed state intact.
- **Command links**
    - **Always present command links in a set of two or more.** Logically, there is no reason to ask a question that has only one answer.
    - **Provide an explicit Cancel button.** Don't use a command link for this purpose. Quite often, users realize that they don't want to perform a task. Using a command link to cancel would require users to read all the command links carefully to determine which one means to cancel. Having an explicit Cancel button allows users to cancel a task efficiently.
    - **If providing an explicit Cancel button leaves a single command link, provide both a command link to cancel and a Cancel button.** Doing so makes it clear that users have a choice. Phrase this command link in terms of how it differs from the first response, instead of just "Cancel" or some variation.
- **"Don't show this again" check boxes**
    - **Consider using a "Don't show this again" option to allow users to suppress a recurring dialog box, only if there isn't a better alternative.** It is better always to show the dialog if users really need it, or simply eliminate it if they don't.
    - **Replace with the specific item.** For example, Don't show this reminder again. When referring to a dialog box in general, use Don't show this message again.
    - **Clearly indicate when user input will be used for future default values** by adding the following sentence under the option: Your selections will be used by default in the future.
    - **Don't select the option by default. If the dialog box really should be displayed only once, do so without asking.** Don't use this option as an excuse to annoy users—make sure the default behavior isn't annoying.
    - **If users select the option and click Cancel, this option does take effect.** This setting is a meta-option, so it doesn't follow the standard Cancel behavior of leaving no side effect. Note that if users don't want to see the dialog in the future, most likely they want to cancel it as well.
- **Links**
    - **Don't assign an** access key. Links are accessed using the Tab key.
    - **Don't add "Click" or "Click here" to the link text.** It isn't necessary because a link implies clicking.
- **Tooltips**

    - **Use tooltips to provide labels for unlabeled controls.** You don't have to give labeled controls tooltips simply for the sake of consistency.
    - **Tooltips may also provide more detail for labeled toolbar buttons if doing so is helpful.** Don't just repeat or give a wordy restatement of what is already in the label.
    - **Avoid covering the object the user is about to view or interact with.** Always place the tip on the side of the object, even if that requires separation between the pointer and the tip. Some separation isn't a problem as long as the relationship between the object and its tip is clear.
        - **Exception:** Full name tooltips used in lists and trees.
    - **For collections of items, avoid covering the next object that the user is likely to view or interact with.** For horizontally arranged items, avoid placing tips to the right; for vertically arranged items, avoid placing tips below.
- **Progressive disclosure**
    - **Use More/Fewer progressive disclosure buttons to hide advanced or rarely used options, commands, and details that users typically don't need.** Don't hide commonly used items, because users might not find them. But make sure whatever is hidden has value.
    - If the surface always displays some options, commands, or details, use the following label pairs:
        - **More/Fewer options.** Use for options or a mixture of options, commands, and details.
        - **More/Fewer commands.** Use for commands only.
        - **More/Fewer details.** Use for information only.
        - **More/Fewer . Use for other object types, such as folders.**
        - **Otherwise:**
            - **Show/Hide options. Use for options or a mixture of options, commands, and details.**
            - **Show/Hide commands. Use for commands only.**
            - **Show/Hide details. Use for information only.**
            - **Show/Hide . Use for other object types, such as folders.**
            - **Progress bars**
                - **Use determinate progress bars for operations that require a bounded amount of time, even if that amount of time cannot be accurately predicted. Indeterminate progress bars show that progress is being made, but provide no other information. Don't choose an indeterminate progress bar based only on the possible lack of accuracy alone.**
                - **Provide a time remaining estimate if you can do so accurately. Time remaining estimates that are accurate are useful, but estimates that are way off the mark or bounce around significantly**

aren't helpful. You may need to perform some processing before you can give accurate estimates. If so, don't display potentially inaccurate estimates during this initial period.
    - Don't restart progress. A progress bar loses its value if it restarts (perhaps because a step in the operation completes) because users have no way of knowing when the operation will complete. Instead, have all the steps in the operation share a portion of the progress and have the progress bar go to completion once.
    - Provide useful progress details. Provide additional progress information, but only if users can do something with it. Make sure the text is displayed long enough for users to be able to read it.
    - Don't combine a progress bar with a busy pointer. Use one or the other, but not both at the same time.
- Prompts
    - Use a prompt when screen space is at such a premium that using a label or instruction is undesirable, such as on a toolbar.
    - The prompt is primarily for identifying the purpose of the text box or combo box in a compact way. It must not be crucial information that the user needs to see while using the control.
    - The prompt text must not be confused with real text. To do this:
        - Draw the prompt text in italic gray and the actual input text in roman black.
        - The prompt text should not be editable and should disappear once users click in or tab into the text box.
            - Exception: If the text box has default input focus, the prompt is displayed, and it disappears once the user starts typing.
    - Don't use ending punctuation or ellipsis.
- Notifications
    - Use notifications for events that are unrelated to the current user activity, don't require immediate user action, and users can freely ignore.
    - Don't abuse notifications:
        - Use notifications only if you need to. When you display a notification, you are potentially interrupting users or even annoying them. Make sure that interruption is justified.
        - Use notifications for non-critical events or situations that don't require immediate user action. For critical events or situations that require immediate user action, use an alternative UI element (such as a modal dialog box).
        - Don't use notifications for feature advertisements!

Keyboard

- Assign initial input focus to the control that users are most likely to interact with first, which is often the first interactive control. If the first interactive control isn't a good choice, consider changing the window's layout.
- Assign tabs stops to all interactive controls, including read-only edit boxes. Exceptions:
  - Group sets of related controls that behave as a single control, such as radio buttons. Such groups have a single tab stop.
  - Properly contain groups so that the arrow keys cycle both forward and backward within the group and stay within the group.
- Tab order should flow from left to right, top to bottom. Generally, tab order should follow reading order. Consider making exceptions for commonly used controls by putting them earlier in the tab order. Tab should cycle through all the tab stops in both directions without stopping. Within a group, tab order should be in sequential order, without exceptions.
- Within a tab stop, the arrow key order should flow from left to right, top to bottom, without exceptions. The arrow keys should cycle through all items in both directions without stopping.
- Present the commit buttons in the following order:
  - OK/[Do it]/Yes
  - [Don't do it]/No
  - Cancel
  - Apply (if present)

where [Do it] and [Don't do it] are specific responses to the main instruction.

- Don't confuse access keys with shortcut keys. While both access keys and shortcut keys provide keyboard access to UI, they have different purposes and guidelines.
- Whenever possible, assign unique access keys to all interactive controls or their labels.Read-only text boxes are interactive controls (because users can scroll them and copy text), so they benefit from access keys. Don't assign access keys to:
  - OK, Cancel, and Close buttons. Enter and Esc are used for their access keys. However, always assign an access key to a control that means OK or Cancel, but has a different label.
- Assign shortcut keys to the most commonly used commands. Infrequently used programs and features don't need shortcut keys

because users can use access keys instead.

- Don't make a shortcut key the only way to perform a task. Users should also be able to use the mouse or the keyboard with Tab, arrow, and access keys.
- Don't assign different meanings to well-known shortcut keys. Because they are memorized, inconsistent meanings for well-known shortcuts are frustrating and error prone.
- Don't try to assign system-wide program shortcut keys. Your program's shortcut keys will have effect only when your program has input focus.

Mouse

- Never require users to click an object to determine if it is clickable. Users must be able to determine clickability by visual inspection alone.
  - Primary UI (such as commit buttons) must have a static click affordance. Users shouldn't have to hover to discover primary UI.
  - Secondary UI (such as secondary commands or progressive disclosure controls) can display their click affordance on hover.
  - Text links should statically suggest link text, and then display their click affordance (underline or other presentation change, with hand pointer) on hover.
  - Graphics links only display a hand pointer on hover.
- Use the hand (or "link select") pointer only for text and graphic links. Otherwise, users would have to click on objects to determine if they are links.

Dialog boxes

- Modal dialog boxes require interaction, so use them for things that users must respond to before continuing with their task. Make sure the interruption is justified, such as for critical or infrequent, one-off tasks that require completion. Otherwise, consider modeless alternatives.
- Modeless dialog boxes don't require interaction, so use them when users need to switch between a dialog box and the owner window. They are best used for frequent, repetitive, or ongoing tasks. However, ribbons, toolbars, and palette windows are often better alternatives.

Property Sheets

- **Make sure the properties are necessary. Don't clutter your property pages with unnecessary properties just to avoid making hard design decisions.**
- **Present properties in terms of user goals instead of technology. Just because a property configures a specific technology doesn't mean that you must present the property in terms of that technology.**
  - **If you must present settings in terms of technology (perhaps because your users recognize the technology's name), include a brief description of the user benefit.**
- **Use specific, meaningful tab labels. Avoid generic tab labels that could apply to any tab, such as General, Advanced, or Settings.**
- **Avoid General pages. You aren't required to have a General page. Use a General page only if:**
  - **The properties apply to several tasks and are meaningful to most users. Don't put specialized or advanced properties on a General page, but you can make them accessible through a command button on the General page.**
  - **The properties don't fit a more specific category. If they do, use that name for the page instead.**
- **Avoid Advanced pages. Use an Advanced page only if:**
  - **The properties apply to uncommon tasks and are meaningful primarily to advanced users.**
  - **The properties don't fit a more specific category. If they do, use that name for the page instead.**
- **Don't use tabs if a property window has only a single tab and isn't extensible. Use a regular dialog box with OK, Cancel, and an optional Apply button instead. However, extensible property windows (which can be extended by third parties) must use tabs.**

Wizards

- **Consider lightweight alternatives first, such as dialog boxes, task panes, or single pages. Wizards are a heavy UI, best used for multi-step, infrequently performed task. You don't have to use wizards—you can provide helpful information and assistance in any UI.**
- **Use Next only when advancing to the next page without commitment. Advancing to the next page is considered a**

commitment when its effect can't be undone by clicking Back or Cancel.

- Use Back only to correct mistakes. Aside from correcting mistakes, users shouldn't have to click Back to make progress in a task.
- When users are committing to a task, use a commit button that is a specific response to the main instruction (for example, Print, Connect, or Start). Don't use generic labels like Next (which doesn't imply commitment) or Finish (which isn't specific) for committing a task. The labels on these commit buttons should make sense on their own. Always start commit button labels with a verb. Exceptions:
  - Use Finish when the specific responses are still generic, such as Save, Select, Choose, or Get.
  - Use Finish to change a specific setting or a collection of settings.
- Use command links only for choices, not commitments. Specific commit buttons indicate commitment far better than command links in a wizard.
- When using command links, hide the Next button, but leave the Cancel button.
- Use Close for Follow-Up and Completion pages. Don't use Cancel because closing the window won't abandon any changes or actions done at this point. Don't use Done because it isn't an imperative verb.
- Don't use "wizard" in wizard names. For example, use "Connect to a Network" instead of "Network Setup Wizard." However, it's acceptable to refer to wizards as wizards. For example: "If you're setting up a network for the first time, you can get help by using the Connect to a Network wizard."
- Preserve user selections through navigation. For example, if the user makes changes, clicks Back, then Next, those changes should be preserved. Users don't expect to have to re-enter changes unless they explicitly chose to clear them.

Wizard pages

- Focus on efficient decision making. Reduce the number of pages to focus on essentials. Consolidate related pages, and take optional pages out of the main flow. Having users click Next completely through your wizard may seem like a good experience at first, but if users never need to change the defaults, the pages are probably unnecessary.

- Don't use Welcome pages—make the first page functional whenever possible. Use an optional Getting Started page only when:
  - The wizard has prerequisites that are necessary to complete the wizard successfully.
  - Users may not understand the purpose of the wizard based on its first Choice page, and there isn't room for further explanation.
  - The main instruction for Getting Started pages is "Before you begin:".
- On pages in which users are asked to make choices, optimize for the most likely cases. These kinds of pages should present actual choices, not just instructions.
  - If you don't use a Getting Started page, explain the purpose of the wizard at the top of the first page of choices.
- Use Commit pages to make it clear when users are committing to the task. Usually the Commit page is the last page of choices, and the Next button is relabeled to indicate the task being committed.
  - Don't use Summary pages that merely summarize the user's previous selections, unless the task is risky (involving security, or loss of time or money) or there is a good chance that users might not understand their selections and need to review them.
- Don't use Congratulations pages that do nothing but end the wizard. If the wizard results are clearly apparent to users, just close the wizard on the final commit button.
  - Use Follow-Up pages when there are related tasks that users are likely to do as follow-up. Avoid familiar follow-up tasks, such as "Send an e-mail message."
  - Use Completion pages only when the results aren't visible and there's no better way to provide feedback for task completion.
  - Wizards that have Progress pages must use a Completion page or Follow-Up page to indicate task completion. For long-running tasks, close the wizard on the Commit page and use notifications to give feedback.

Error messages

- Don't give error messages when users aren't likely to perform an action or change their behavior as the result of the message. If there is no action users can take, or if the problem isn't significant, suppress the error message.

- Whenever possible, propose a solution so users can fix the problem. However, make sure the proposed solution is likely to solve the problem. Don't waste users' time by suggesting possible, but improbable, solutions.
- Be specific. Avoid vague wording, such as syntax error and illegal operation. Provide specific names, locations, and values of the objects involved.
- Don't use phrasing that blames the user or implies user error. Avoid using you and your in the phrasing. While the active voice is generally preferred, use the passive voice when the user is the subject and might feel blamed for the error if the active voice were used.
- Don't use OK for error messages. Users don't view errors as being OK. If the error message has no direct action, use Close instead.
- Don't use the following words:
  - Error, failure (use problem instead)
  - Failed to (use unable to instead)
  - Illegal, invalid, bad (use incorrect or not valid instead)
  - Abort, kill, terminate (use stop instead)
  - Catastrophic, fatal (use serious instead)

These terms are unnecessary and contrary to the encouraging tone of Windows. Instead, an error icon, when used correctly, sufficiently communicates that there is a problem.

- Don't accompany error messages with sound effects. Doing so is jarring and unnecessary.

Warning messages

- Warnings describe a condition that might cause a problem in the future. Warnings aren't errors or questions, so don't phrase routine questions as warnings.
- Don't give warning messages when users aren't likely to perform an action or change their behavior as the result of the message. If there is no action users can take, or if the condition isn't significant, suppress the warning message.

Confirmations

- Don't use unnecessary confirmations. Use confirmations only when:

    - There is a clear reason not to proceed and a reasonable chance that sometimes users won't.
    - The action has significant consequences or cannot be easily undone.
    - The action has consequences that users might not be aware of.
    - Proceeding with the action requires users to make a choice that doesn't have a suitable default.
    - Given the current context, users are likely to have performed an action in error.
- Phrase confirmations as a yes or no question, and provide yes or no answers. Unlike other types of dialog boxes, confirmations are designed to prevent users from making quick decisions. If users don't put thought into their response, a confirmation has no value.

Icons

- All icons should adhere to the **Aero-style icon guidelines**. Replace all Windows XP-style icons.
- Choose standard icons based their message type, not the severity of the underlying issue:
    - Error. An error or problem that has occurred.
    - Warning. A condition that might cause a problem in the future.
    - Information. Useful information.

If an issue combines different message types, focus on the most important aspect that users need to act on.

- Icons must always match the main instruction or other corresponding text.
- Error icons aren't needed for non-critical user input problems. However, icons are needed for in-place errors, because otherwise such contextual feedback would be too easy to overlook.
- Don't use warning icons to "soften" non-critical errors. Errors aren't warnings; apply the error icon guidelines instead.
- For question dialogs, use warning icons only for questions with significant consequences. Don't use warning icons for routine questions.

Help

- Link to specific, relevant Help topics. Don't link to the Help home page, the table of contents, a list of search results, or a page that

just links to other pages. Avoid linking to pages structured as a large list of frequently asked questions, because it forces users to search for the one that matches the link they clicked. Don't link to specific Help topics that aren't relevant and helpful to the task at hand. Never link to empty pages.

- Don't put Help links on every window or page for the sake of consistency. Providing a Help link in one place doesn't mean that you have to provide them everywhere.
- Whenever possible, phrase Help links text in terms of the primary question answered by the Help content. Don't use the "Learn more about" or "Get help with this" phrasing.
- Use the entire Help link for the link text, not just the keywords.
- Use complete sentences.
- Don't use ending punctuation or ellipses, except for question marks.
- If the Help content is online, make that clear in the link text. Doing so helps make the result of the links predictable.

Is this page helpful?

Yes No

English (United States) Theme

Previous Version Docs Blog Contribute Privacy & Cookies

Terms of Use Site Feedback Trademarks © Microsoft 2020