RESOURCE AND PATIENT MANAGEMENT SYSTEM

GUI Programming Standards and Conventions

Version 1.0
October 2010

Office of Information Technology (OIT)
Division of Information Resource Management
Albuquerque, New Mexico

Table of Contents

1.0 Purpose, Policy, and Standards and Conventions

1.1 Purpose

The purpose of this document is to provide a set of standards and conventions for graphical user interface (GUI) development. They assume an understanding of and compliance with the user interface guidelines set forth by Microsoft Corporation (submitted as an addendum with this document).

1.2 Policy

1.2.1 Conformance

All GUI software developed for IHS will conform to the GUI Standards and Conventions. In areas where standards are not specified, software development will conform to other applicable IHS programming standards and conventions and industry standards, unless an exemption is requested and approved by the Standards and Conventions (SAC) committee.

2.0 GUI Programming Standards and Conventions

All GUI Indian Health Service (IHS) software will meet the following standards and comply with the spirit of the conventions.

2.1 Acceptable Icons

2.1.1 Icons–Application and Application Launch Level

Icons for application and application launch (to include but not limited to Mac, personal computer [PC], iPhone, iPad, Android, or other mobile device) shall be graphics- or symbols-free of representing any culture, race or ethnicity. Icons must not contain any definitive marks or symbols or detail that may be used to potentially depict a specific gender, culture, nationality, tribe, or race. An example of an acceptable icon for an application that tracks personal health information is shown in Exhibit 2A. Notice it is a nondescriptive icon with no detailed facial features (i.e. eyes, ears, nose; Exhibit 2B).

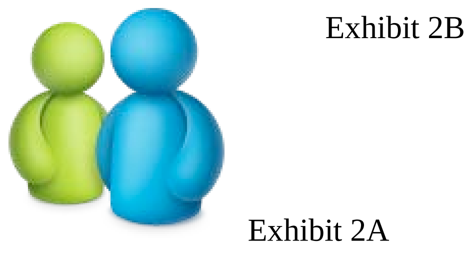


Figure 2-1: Nondescriptive icons

3.0 GUI Programming Standards–UI Document Addendum

What follows and incorporated is a rule list incorporated into this document from Windows User Experience Guide (the guide can be downloaded from http://msdn.microsoft.com/en-us/library/aa511258.aspx).

3.1 Custom Control (Beginning on Page 29)

- Use custom controls for unusual behaviors that aren't supported by the common controls.
- Ensure custom controls support system metrics and colors and respect all user changes to these settings.
- Ensure custom controls conform to the Windows **accessibility** guidelines.
- Use 3-D graphics to help users visualize, examine, and interact with three-dimensional objects, charts, and graphics.
- Make sure there are clear user scenarios that support the need for 3-D graphics features.

3.1.1 Custom Controls: Animations

- Use illustration animations that have a single interpretation. They have little value if confusing.
- Show one thing at a time to avoid overwhelming users.
- Play at the optimal speed—not so fast they are difficult to understand, but not so slow they are tedious to watch.
- Gradually increase the speed of repeated animations. Viewers will already be familiar with the animation, so increasing speed slowly will feel right.
- Use timing to emphasize importance, such as slowing down for important parts.
- Use effect animations for objects that the user is currently interacting with. Such animations aren't distracting because the user is already focused on the object.
  - Minimize use of effect animations that show status. Make sure:
    - They have real value by providing additional information users can actually use. Examples include transient status changes and emergencies.
    - They are subtle.
    - They are short in duration and therefore not running most of the time.
    - They can be turned off.

  - Keep effect animations low-key so they don't draw too much attention to themselves. Avoid movement or use small movements, but prefer fades and changes in overlays.
- Must start or end with the selected object. Don't show relationships between objects the user isn't currently interacting with
- Must complete within a half-second or less.

3.1.2 Transition Animations

- Use to show relationships between states. Animating state changes makes them easier to understand and appear smoother.
- Make sure transitions have natural mappings. For example, an opening window transition should be upward and expand; a closing window transition should be downward and contract.
- Must complete within a half-second or less.

3.1.3 Feedback Animations

- Must have clearly identifiable completion and failure states.
- Must stop showing progress when the underlying process isn't making progress.

3.1.4 Progressive Disclosure

- Make sure the progressive disclosure mechanism is visible at all times.
- Use the appropriate glyph. Use double or single chevrons for surfaces that slide open to show the remaining items in hidden content.
- Point the glyph in the right direction. Chevrons point in the direction where the action will occur.

3.2 Direct Manipulation (Page 32)

- Make direct manipulation visible by: Changing the pointer to a hand on mouseover.
- Showing drag handles on object selection.
- Showing movement when an object is dragged, then show appropriate drop targets. Also, clearly show when an object is successfully dropped or when the drop is cancelled.
- Showing an in-place text box on double selection for renaming.
- Showing most useful properties with tooltips.

- Prevent accidental manipulation by: Locking by default objects that are crucial or likely to be manipulated accidentally. Provide a way to unlock in the object's context menu.
- Providing an obvious way to undo accidental manipulations.
- Make direct manipulation accessible by always providing alternatives.

3.2.1 Hosting in Browser:

Host your program in the browser to navigate from a Web page to your program.

3.3 Balloons (Page 40)

- Display the balloon as soon as the problem or special condition is detected, even if repeatedly, without any noticeable delay.
- For problems involving individual characters or the maximum input size, display the balloon immediately on input.
- For problems involving the input value (including requiring a non-blank value), display the balloon when the owner control loses input focus. Otherwise, displaying balloons while users are entering potentially valid input can be distracting and annoying.
- Display only one balloon at a time. Displaying multiple balloons can be overwhelming. If a single event results in multiple problems, either present all the problems at once or report only the most important problem.
- Remove a balloon when:
  - The problem is resolved or special condition is removed.
  - The user enters valid data (for input problems).
- The balloon times out. By default balloons remove themselves after 10 seconds, although users can change this by modifying the SPI_MESSAGEDURATION system parameter.
- **Remove the timeout if users can't continue until the problem is resolved. Developers:** In Win32, you can set the display time with the TTM_SETDELAYTIME message.
- **Display balloons below their owner control.** Doing so allows users to view the context, including the owner control and its label. Microsoft® Windows® automatically adjusts balloon positions so that they are completely on screen. The default behavior is to display a balloon above its owner control, as done with notifications.

3.3.1 Passwords and PIN

- Use a balloon to indicate that Caps Lock is on.

- **Use a balloon to indicate when users attempt to exceed the maximum input size.** Reaching the maximum input size is much less obvious in password and PIN text boxes than ordinary text boxes.
- **Use a balloon to indicate when users input incorrect characters.** However, it is better not to have such restrictions because they reduce the security of the password or PIN. To prevent information disclosure, the balloon should mention only documented facts about valid passwords or PINs.
- **When users click a balloon, just dismiss the balloon without displaying any other UI or having any other side effect.** Unlike notifications, balloons shouldn't have close buttons.

3.3.2 Balloons Accessibility

When used properly, balloons enhance accessibility. For balloons to be accessible:

- Only display balloons that relate to the user's current activity.
- Keep the balloon text concise. Doing so makes the balloon text easier to read for users with low vision, and minimizes the interruption when read by screen readers.
- Redisplay the balloon whenever the problem or condition recurs.

3.3.3 Title Text

- Use title text that briefly summarizes the input problem or special condition in clear, plain, concise, specific language. Users should be able to understand the purpose of the balloon quickly and with minimal effort.
- Use text fragments or complete sentences without ending punctuation.
- Use sentence-style capitalization.
- Use no more than 48 characters (in English) to accommodate localization. The title has a maximum length of 63 characters and must be able to expand by at least 30% to accommodate localization.

3.3.4 Body Text

- Use the first sentence of the body text to state the problem or condition in a way that is clearly relevant to the user. Don't repeat the information in the title. Omit this if there is nothing more to add.
- Use the second sentence to state what the user can do to resolve the problem or revert the state. In accordance with the Style and Tone guidelines, there's no need to use the word "please" in this statement. Put two line breaks between the first and second sentences.

- Explain how to resolve the problem or revert the state even if that explanation is obvious, but omit redundancy between the problem statement and its resolution. Exceptions:
  - Omit the resolution if it can't be expressed concisely or without significant redundancy.
  - Omit the resolution if there is nothing for the user to do, such as when incorrect characters are ignored.
  - **Use complete sentences with ending punctuation.**
  - **Use sentence-style capitalization.**
  - **Use no more than 200 characters (in English) to accommodate localization.** The body text has a maximum length of 255 characters and must be able to expand by at least 30% to accommodate localization.

3.3.5 Documentation

When referring to balloons:

- Use the exact title text, including its capitalization.
- Refer to the component as a *balloon*, not as a *notification* or an *alert*.

When possible, format the title text using bold text. Otherwise, put the title in quotation marks only if required to prevent confusion.

3.3.5.1 Is This The Right Control?

To decide, consider these questions:

- Is the check box used to toggle an option on or off or to select or deselect an item? If not, use another control.
- Are the selected and cleared states clear and unambiguous opposites? If not, use radio buttons or a drop-down list so that you can label the states independently.
- When used in a group, does the group comprise independent choices, from which users may choose zero or more? If not, consider controls for dependent choices, such as radio buttons and check box tree views.
- When used in a group, does the group comprise dependent choices, from which users must choose one or more? If so, use a group of check boxes and handle the error when none of the options are selected.
- Is the number of options in a group 10 or fewer? Since the screen space used is proportional to the number of options, keep the number of check boxes to 10 or fewer. For more than 10 options, use a check box list.
- Would a radio button be a better choice? Where check boxes are suitable only for turning an option on or off, radio buttons can be used for completely different options. If both solutions are possible:

  - Use radio buttons if the meaning of the cleared check box isn't completely obvious.
  - Use radio buttons on wizard pages to make the alternatives clear, even if a check box is otherwise acceptable.
  - Use radio buttons if you have enough screen space and the options are important enough to be a good use of that screen space. Otherwise, use a check box or a drop-down list.
  - Use a check box if there other check boxes on the window.
  - **Does the option present a program option, rather than data?** The option's values shouldn't be based on context or other data. For data, use a check box list or **multiple-selection list**.
- **Group related check boxes**. Combine related options and separate unrelated options into groups of 10 or fewer, using multiple groups if necessary.
- **List check boxes in a logical order**, such as grouping highly related options together or placing most common options first, or following some other natural progression. Alphabetical ordering isn't recommended because it is language dependent, and therefore not localizable.
- **Align check boxes vertically, not horizontally**. Horizontal alignment is harder to read.
- **Don't use the mixed state to represent a third state.** The mixed state is used to indicate that an option is set for some, but not all, child objects. Users shouldn't be able to set a mixed state directly—rather the mixed state is a reflection of the child objects. The mixed state isn't used as a third state for an individual item. To represent a third state, use radio buttons or a drop-down list instead.
- **Clicking a mixed state check box should cycle through all selected, all cleared, and the original mixed states.** For forgiveness, it's important to be able to restore the original mixed state because the settings might be complex or unknown to the user. Otherwise, the only way to restore the mixed state with confidence would be to cancel the task and start over.
- **Don't use check boxes as a progress indicator**. Use a **progress indicator** control instead.
- **Show disabled check boxes using the correct selection state.** Even though users can't change them, disabled check boxes convey information so they should be consistent with results.
  - Don't use the selection of a check box to: Perform commands.
  - Display other windows, such as a dialog box to gather more input.
  - Dynamically display other controls related to the selected control (screen readers cannot detect such events).

3.3.6 Don’t Show This <item> Again

Consider using a *Don’t show this <item> again* option to allow users to suppress a recurring dialog box only if there isn’t a better alternative. Try to determine beforehand if users really need the dialog; if they do, always show the dialog, and if they don’t, eliminate the dialog.

3.3.7 Subordinate Controls

- Place subordinate controls to the right of or below (indented, flush with the check box label) the check box and its label. End the check box label with a colon.
  - **Leave dependent editable text boxes and drop-down lists enabled if they share the check box’s label.** When users type or paste anything into the box, select the corresponding option automatically. Doing so simplifies the interaction.
  - If you nest check boxes with radio buttons or other check boxes, **disable these subordinate controls until the high-level option is selected**. Doing so avoids confusion about the meaning of the subordinate controls.
  - Make subordinate controls to a check box contiguous with the check box in the tab order.
  - **If selecting an option implies selecting subordinate check boxes, explicitly select those check boxes to make the relationship clear.**
  - **Use dependent check boxes if the alternatives add unnecessary complexity**. While check boxes should be independent options, sometimes alternatives such as radio buttons add unnecessary complexity.

3.3.8 Default Values

If a check box is for a user option, **set the safest (to prevent loss of data or system access), most secure and private state by default.** If safety and security aren’t factors, select the most likely or convenient value.

3.3.9 Labels

3.3.9.1 Check Box Labels

- Label every check box.
- Assign a unique **access key** to each label. For guidelines, see **Keyboard**.
- Use **sentence-style capitalization**.
- Write the label as a phrase or an imperative sentence, and use no ending punctuation. **Exception:** If a check box label also labels a subordinate control that follows it, end the label with a colon.
- Write the label so that it describes the selected state of the check box.

  - For a group of check boxes, use parallel phrasing and try to keep the length about the same for all labels.
  - For a group of check boxes, focus the label text on the differences among the options. If all the options have the same introductory text, move that text to the group label.
  - Use positive phrasing. Don’t phrase a label so that selecting a check box means not to perform an action. **Exception: Don’t show this <item> again** check boxes.
  - Describe just the option with the label. Keep labels brief so it’s easy to refer to them in messages and documentation. If the option requires further explanation, provide the explanation in a **static text** control using complete sentences and ending punctuation.
  - If an option is strongly recommended, consider adding “(recommended)” to the label. Be sure to add to the control label, not the supplemental notes.
  - If you must use multiline labels, align the top of the label with the check box.
  - Don’t use a subordinate control, the values it contains, or its units label to create a sentence or phrase. Such a design isn’t localizable because sentence structure varies with language.

3.3.9.2 Check Box Group Labels

- Use the group label to explain the purpose of the group, not how to make the selection. Assume that users know how to use check boxes. For example, don’t say “Select any of the following choices”.
- End each label with a colon.
- Don’t assign an access key to the label. Doing so isn’t necessary and it makes the other access keys harder to assign.
- For a selection of one or more dependent choices, explain the requirement on the label.

3.4 Command Buttons (Page 54)

- Is the command button used to initiate an immediate action? If not, use another control.
- Would a link be a better choice? Use a link if: The action is to navigate to another page, window, or Help topic. Exception:
  - Wizard navigation uses Back and Next command buttons.
  - The command is embedded in a body of text.
  - The command is secondary in nature. That is, it does not relate to the primary purpose of the window. In this case, either a lightweight command button or link would be appropriate.

  - The command is part of a menu or group of related links.
  - The label is lengthy, consisting of five or more words, thus giving a command button an awkward appearance.
- **Would a combination of radio buttons and generic command buttons be a better choice?** Often radio buttons are used in conjunction with generic command buttons (OK, Cancel) in place of a set of specific command buttons when any of the following are true:
  - There are five or more possible actions.
  - Users need to view additional information before making a decision.
  - Users need to interact with the choices (perhaps to see additional information) before making a decision.
  - Users view the choices as options instead of different commands.

3.4.1 Using Ellipses

While command buttons are used for immediate actions, more information might be needed to perform the action. Indicate a command that needs additional information (including confirmation) by adding an ellipsis at the end of the button label.

This doesn't mean you should use an ellipsis whenever an action displays another window—only when additional information is required to perform the action. Consequently, any command button whose implicit verb is to "show another window" doesn't take an ellipsis, such as with the commands About, Advanced, Help (or any other command linking to a Help topic), Options, Properties, or Settings.

3.4.2 Command Button Usage

- Display a busy pointer if the result of clicking a command button isn't instantaneous. Without feedback, users might assume that the click didn't happen and click again.
- If the same command button appears in more than one window, try to use the same label text and access key, and locate it in approximately the same place in each window when practical.
- For command buttons with text labels, use a minimum button width and the standard command button height. For more information, see Recommended sizing and spacing.
- For each window make the command buttons the same width. If that's impractical, limit the number of different widths for command buttons with text labels to two.
  - When another control interoperates with a command button, such as a text box with a Browse button, **denote the relationship by placing the command button in one of three places:**

    - To the right of and top-aligned with the other control.
    - Below and left-aligned with the other control.
    - Vertically centered between controls that interoperate (such as Add and Remove buttons between two interoperating list boxes).

- If multiple command buttons interoperate with the same control, **vertically stack them to the right of and top-aligned with the other control, or horizontally place them left-aligned under the control.**
- When command buttons are subordinate to other controls, **use the above placement and disable the subordinate command button until the superior control is selected.**
- **Don’t use narrow, short, or tall command buttons with text labels** because they tend to look unprofessional. Try to work with the default widths and heights.

Avoid combining text labels and graphics on command buttons. Combining text and graphics usually adds unnecessary visual clutter and does not improve the user’s comprehension. Consider combining text and graphics only when the graphic aids in comprehension, such as when it is a standard symbol for the command or it helps users visualize the results of the command. Otherwise, prefer text, but use either text or graphics.

Don’t use command buttons to set state. Use radio buttons or check boxes instead. Command buttons are only for initiating actions.

3.4.3 Split Buttons

- **Make the most likely command the default behavior**. If there is more than one likely command, choose one that doesn’t require additional information.
- **If the most likely command is the last user selection, change the button label to the last selection.**
- **Display the default command using bold text in the menu**. Doing so makes it easier for users to find the default command, especially when the default command is dynamic or the split button uses a graphic instead of a text label.

3.4.4 Default Values

- Include a default command button on every dialog box. **Select the safest (to prevent loss of data or system access) and most secure command to be the default**. If safety and security aren’t factors, select the most likely or convenient command.
- **Don’t make a destructive action the default command button** unless there is an easy way to undo the command.

3.4.5 Recommended Sizing And Spacing

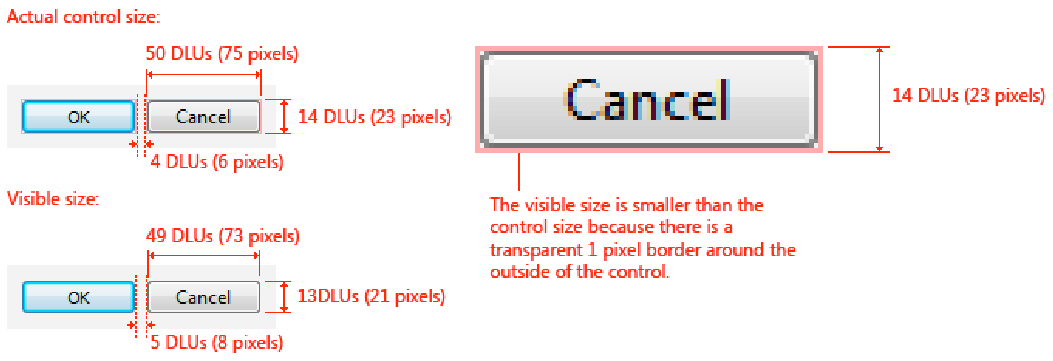


Figure 3-1: Recommended size and spacing, in pixels, of buttons.

3.4.6 Labels

- Label every command button.
- If the button has a graphic label only, assign its Name property to an appropriate text label. This enables assistive technology products such as screen readers to provide users with alternative information about the graphic.
  - **Exceptions:**
    - Don’t assign access keys to OK and Cancel buttons, because Enter is the access key for the default button (which is usually the OK button), and Esc is the access key for Cancel. Doing so makes the other access keys easier to assign.
    - Don’t assign access keys to short browse buttons (labeled “...”), because they can’t be assigned uniquely.
  - 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.
    - **Exception:** The following standard labels are acceptable without verbs: Advanced, Back, Details, Forward, Less, More, New, Next, No, OK, Options, Previous, Properties, Settings, and Yes.
  - While short labels are preferred, use enough text to explain the command sufficiently. Use a direct object (a noun after the verb) when the object is not apparent from context. Ideally users shouldn’t have to read anything else to understand the label.

  - Use **sentence-style capitalization**. Doing so is more appropriate for **Windows tone** and the use of short phrases for command buttons. **Exception:** For legacy applications, you may use **title-style capitalization** if necessary to avoid mixing capitalization styles.
- Don't use *later* in command button labels if it implies an action that won't happen. For example, don't use *Install later* (in contrast to *Install now*) unless that command installs at a later time. Instead, use either *Don't install* or *Cancel*.
  - Use an Advanced button only for options that are relevant to advanced users or require advanced user knowledge.
  - Don't use an Advanced button for features that are considered technologically advanced. For example, a printer's stapling feature is not an advanced option, but its color management system is
    - For command buttons that open other windows, choose a label that uses part or all of the secondary window's title bar text.

      (Page 62) For example, a command button labeled *Browse* might open a dialog box entitled *Browse for Folder*. Using the same terminology throughout the task helps to keep users oriented.

- When asking a question, use labels that match the question. Don't use OK/Cancel to answer Yes/No questions.
- Don't use an ellipsis when the successful completion of the action is simply to display another window. The following commands never take an ellipsis: About, Advanced, Options, Properties, Help.
- In case of ambiguity (for example, the command label lacks a verb), decide based on the most likely user action. If simply viewing the window is a common action, don't use an ellipsis.
- For browse buttons, use short browse buttons (labeled "...") when there are more than two browse buttons in a window. Always use the short version when you want to display a browse button in a grid.
- For directional buttons, use a single angle bracket and have it point in the direction where the action takes place.
- Use command buttons for: Primary commands.
- Displaying windows used to gather input or making choices, even if they are secondary commands.
- Destructive or irreversible actions, as well as commitment within wizards and page flows.
  - Use links for navigation to another page, window, or Help topic; display definitions; and command menus.
  - Consider using links to deemphasize secondary commands.

  - Use short command button labels, consisting of four or fewer words. Links can have longer labels.

3.4.7 Command Links

- **Are the options responses to the main instruction and related to the primary purpose of the window or page?** Must users respond to them to do something other than just navigating to a different page? If not, use another control such as command buttons or links. Command links aren't appropriate for secondary or optional choices, or pure navigation.
- **Is the control used to choose one response from a set of mutually exclusive responses?** If not, use another control. To let users choose individual commands, use command buttons or links.
- **For dialog boxes, does clicking the control close the window?** If not, use a control that doesn't require closing the window, such as radio buttons, command buttons, or links.
- **For wizards and page flows, does clicking advance to the next page without commitment?** Don't use command links to commit to a task; use commit buttons instead. Because command links look like links and users associate links with navigation within a page flow, links aren't appropriate for **Commit pages** because users should always be able to back out.
- **For wizards and page flows, are other pages using command links?** If so, and all other factors being equal, prefer command links for consistency across pages.
- **Is the number of responses between two and five?** There should never be a single command link. Because command links are large controls and the screen space used is proportional to the number of options, keep the number of responses to five or fewer. For six or more options, use radio buttons, regular links, or a single-selection **list view**.

Would a combination of radio buttons and a commit button be a better choice? Radio buttons are a better choice when any of the following are true:

- There is a strong default option that you want most users to select. Users are less likely to change a default radio button than a default command link—especially in a wizard, where users are accustomed to clicking Next to accept appropriate defaults. On the other hand, command links are a better choice if you want to encourage users to make an explicit choice.
- Users need to interact with the choices (perhaps to see additional information) before making a decision. For example, selecting a radio button might display a description about the option dynamically. *In this example, selecting a radio button displays a description of the option.*
- There are secondary or related options on the page. Command links tend to dominate the page, making it easy to overlook everything else. Furthermore, once a command link is clicked, it's impossible to select secondary options.

- **There are secondary or related options on the page.** Command links tend to dominate the page, making it easy to overlook everything else. Furthermore, once a command link is clicked, it's impossible to select secondary options.

For dialog boxes, would a group of commit buttons be a better choice? Command links work better when the options require longer, more explanatory responses and supplemental explanations, but a group of commit buttons is a better choice if there are a few simple options.

Note: Command links require Windows Vista® or later, so they aren't suitable for earlier versions of Windows. You can use regular links as a substitute.

3.4.7.1 Command Link Design

We can simplify this dialog box by applying three command link guidelines:

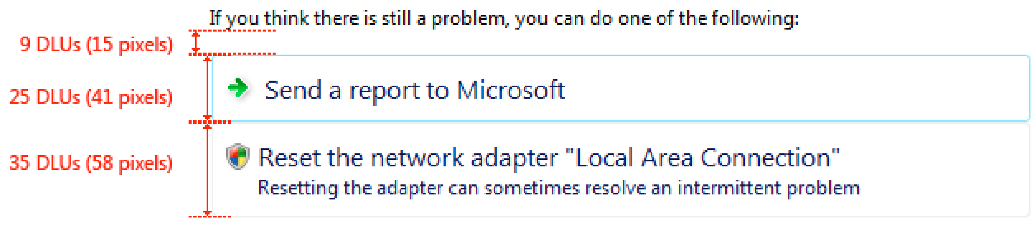
- **Don't use a supplemental explanation that is a wordy restatement of the command link.** Use a supplemental explanation only when you can't make a command link self-explanatory. Providing a supplemental explanation for one command link doesn't mean that you have to provide them for all commands.
- **Select the safest (to prevent loss of data or system access) and most secure response to be the default.** If safety and security aren't factors, select the most likely or convenient response.
- **Provide an explicit Cancel button.** Don't use a command link for this purpose.
- **Display a busy pointer if the result of clicking a command link isn't instantaneous.** Without feedback, users might assume that the click didn't happen and click again.
  - **Present the most commonly used command links first.** The resulting order should roughly follow the likelihood of use, but also have a logical flow. **Exception:** Command links that result in doing everything should be placed first.
  - **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 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.
  - **Use Close instead of Cancel if you can't return the environment to its previous state, leaving no side effect.**

- **Don't display disabled command links.** If a command link doesn't apply to the current context, remove it instead. If removing all the command links that don't apply leaves a single command link, either eliminate the window or page, or display a **confirmation** if explicit user consent is needed.

- **All command links need an icon.** The icons help users distinguish command links from regular links and user interface text.
- **Use the arrow icon only for command links.** Regular links shouldn't use the arrow icon unless they are being used as a substitute for command links in Windows XP.
- **Use the security shield icon to indicate that a response requires immediate elevation.** For additional guidelines on using the security shield icon, see the **User Account Control**.
- **Use custom icons only if they help users visually identify and differentiate the options.** Don't use custom icons if they aren't immediately recognizable or meaningful.
- **For custom icons, use 16 × 16 or 32 × 32 pixel icons.** Use the larger icons if there is sufficient space and they benefit visually from the larger size. If you need security shield overlays, use 32 × 32 or 48 × 48 pixel icons.
- **Avoid mixing custom icons with the standard arrow icon on a window or a page.** If you use a custom icon on a surface, try to use all custom icons. However, prefer the standard arrow icon over meaningless custom icons.

3.4.7.2 Default Values

- Select the safest (to prevent loss of data or system access) and most secure response to be the default. If safety and security aren't factors, select the most likely or convenient response.
- When practical, make the first response the default option because users often expect that—unless that order isn't logical.
- For dialog boxes, don't make a destructive action the default command link unless there is an easy way to undo the action.



If you think there is still a problem, you can do one of the following:

Send a report to Microsoft

Reset the network adapter "Local Area Connection"
Resetting the adapter can sometimes resolve an intermittent problem

Figure 3-2: Pixel ranges for text in a dialog box

3.4.7.3 Command Link Labels

- **Choose a concise label that clearly communicates and differentiates what the command link does.** It should be self-explanatory and correspond to the main instruction. Focus the labels on the differences among the responses. Users shouldn't have to figure out what the command link really means or how it differs from other command links.
  - **Focus command link labels on helping users make the right decision.** Omit details that don't affect the choice. The labels don't have to be a complete specification of what will happen.
  - **Start command links with a verb.** Don't use *click*, however, because the label should communicate what the command link does, not how it works. **Exception:** If all the command links begin with the same verb or phrase, eliminate the redundant verb or phrase.
    - In general, **use positive phrasing** (providing a choice to do something). Negative phrasing (providing a choice not to do something) is acceptable if it makes the labels easier to understand.
    - **Use parallel phrasing and single line labels.** Long labels discourage reading and shouldn't be necessary. Also, moderately sized labels are easier to refer to in documentation.
    - **Use sentence-style capitalization.**
    - **Don't use ending punctuation unless the label is a question.**
    - **Assign a unique access key.** For guidelines, see **Keyboard**.
    - **Don't use ellipses.** Ellipses mean that more information might be needed to perform the action. Properly used command links don't need ellipses because they have an immediate effect.
    - **If a response is strongly recommended, add "(recommended)" to the label.** Be sure to add to the label, not the supplemental explanation.
    - **If a response is intended only for advanced users, consider adding "(advanced)" to the label.** Be sure to add to the label, not the supplemental explanation.

3.4.7.4 Supplemental Explanations

- If a command link requires further explanation, **provide a supplemental explanation**. Supplemental explanations describe why users might want to choose a response or what happens if a response is chosen.
  - **Don't use a supplemental explanation that is wordy restatement of the command link.** Use a supplemental explanation only when you can't make a command link self-explanatory. Providing a supplemental explanation for one command link doesn't mean that you have to provide them for all.

- **Focus supplemental explanations on helping users make the right decision.** Omit details that don't affect the choice. The supplemental explanations don't have to be a complete specification of what will happen.
- **Use parallel phrasing and at most three lines of text.** Long supplemental explanations discourage reading and shouldn't be necessary.
- **Use complete sentences and ending punctuation.**

3.4.7.5 Command Link Group Labels

Don't use group labels. Main instructions act as the group label for command links.

3.4.8 Dropdown List and Combo Boxes

- Is the control used to choose one option from a list of mutually exclusive values? If not, use another control. To choose multiple options, use a standard multiple-selection list, check box list, list builder, or add/remove list instead.
- Are the options commands? If so, use a menu button or split button instead. Use drop-down lists and combo boxes for objects (nouns) or attributes (adjectives), but not commands (verbs).
- Does the list present data, rather than program options? Either way, a drop-down list or combo box is a suitable choice. By contrast, radio buttons are suitable only for a small number of program options.

3.4.8.1 Drop-Down Lists

- **Is there a default option that is recommended for most users in most situations?** Is seeing the selected option far more important than seeing the alternatives? Consider using a drop-down list if you don't want to encourage users to make changes by hiding the alternatives. If not, consider radio buttons, a single-selection list, or an editable list box, which give more emphasis to the alternative choices.
  - **Do you want to draw attention to the option?** If so, consider radio buttons, a single-selection list, or an editable list box, which tend to draw more attention by taking more screen space. Because drop-down lists are compact, they are good choices for options that you want to underemphasize.
  - **Is screen space at a premium?** If so, use a drop-down list because the screen space used is fixed and independent of the number of choices.
  - **Are there other drop-down lists on the window?** If so, consider using a drop-down list for consistency.

3.4.8.2 Editable Drop-Down Lists

In addition to the principles just provided for drop-down lists, the following also apply:

- **Are the possible choices constrained?** If so, use a normal drop-down list instead. Combo boxes are for unconstrained input, in which users may need to enter a value not currently in the list. Because the input is unconstrained, if users enter text that isn't valid you must handle the error with an error message.
- **Can you enumerate the most likely choices in advance**? If not, use a text box instead.
- **Is the drop-down list being used to list previous user input?** Unless users need to review the complete list of previous input, use a text box with the auto-complete option instead.
- **Will users need assistance in selecting valid values?** If so, use a text box with a **Browse button** instead.
- **Is it important to encourage users to review the alternative choices or invite change?** If so, consider using an editable list box instead. With an editable drop-down list, users aren't going to be aware of the alternatives until the list is dropped.
- **Do users need to locate an item rapidly in a large list?** (Win32 only) If so, use a combo box because users can select an item by typing its full name. By contrast, the Win32 drop-down list selects items based only by the last character typed (so typing "Jun" into a list of months would match November, not June). In this case, use a combo box even if the possible choices are constrained.

3.4.9 Editable List Boxes

- **Are the possible choices constrained?** If so, use a single-selection list or normal drop-down list instead. Combo boxes are for unconstrained input, where users may need to enter a value not currently in the list. Because the input is unconstrained, if users enter text that is not valid you must handle the error with an error message.
- **Can you enumerate the most likely choices in advance?** If not, use a text box instead.
- **Is it important to encourage users to review the alternative choices or invite change?** If not, consider an editable drop-down list instead.
- **Do you want to draw attention to the option?** If not, consider an editable drop-down list instead. Because drop-down lists are compact, they are good choices for options that you want to underemphasize.
- **Is screen space at a premium?** If so, use an editable drop-down list because the screen space used is fixed and independent of the number of choices.
- **Always include at least three items in editable list boxes to justify the additional screen space.**

3.4.10 Usage Patterns

3.4.10.1 Guidelines

3.4.10.1.1 General

Don’t use the change of a drop-down list or combo box to: Perform commands.

Display other windows, such as a dialog box to gather more input.

Dynamically display other controls related to the selected control (screen readers cannot detect such events).

3.4.10.2 Presentation

- **Sort list items in a logical order**, such as grouping highly related options together, placing most common options first, or using alphabetical order. Sort names in alphabetical order, numbers in numeric order, and dates in chronological order. Lists with 12 or more items should be sorted alphabetically to make items easier to find.
- **Place options that represent All or None at the beginning of the list, regardless of the sort order of the remaining items.**
- **Enclose metaoptions in parentheses.**
- **When disabling a drop-down list or combo box, also disable any associated labels and command buttons.**

3.4.11 Drop-Down Lists

- When a single drop-down list is used to change the view of an associated control, **change the view immediately on selection instead of requiring a separate command button.** Use a separate command button only if the list takes a significant amount of time to render. However, list headers and **menu buttons** are the preferred controls for this purpose.
- **Don’t have blank list items—use metaoptions instead**. Users don’t know how to interpret blank items, whereas the meaning of metaoptions is explicit.

3.4.11.1 Preview Drop-Down Lists

- Use previews in the list items when it is better to show with images than describe using text alone.
  - Don’t use unnecessary, unhelpful icons in previews.

3.4.12 Combo Boxes

- **Limit the length of the input text when you can.** For example, if the valid input is a number between 0 and 999, use a combo box that is limited to three characters.
- **If there are many possible options, focus the list contents on the most likely options**. Because users can enter values that aren't in the list, combo boxes don't have to list all choices, just the likely choices or a representative sample.

3.5 Default Values (Page 87)

Select the safest (to prevent loss of data or system access) and most secure option by default. If safety and security aren't factors, select the most likely or convenient option. **Exception:** Display a blank default value if the control represents a property in a **mixed state**, which happens when displaying a property for multiple objects that don't have the same setting.

3.5.1 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 list in a compact way. It must not be crucial information that users need to see while using the combo box.

Don't use prompts just to direct users to select something from the list or to click buttons. For example, prompts like *Select an option* or *Enter a filename and then click Send* are unnecessary.

- Draw the prompt text in italic gray and real text in normal black. The prompt text must not be confused with real text.
- Keep the prompt text concise. You can use fragments instead of full sentences.
- Use **sentence-style capitalization**.
- Don't use ending punctuation or ellipsis.
  - The prompt text should not be editable, and should disappear once users click in or tab into the text box. **Exception:** The prompt is displayed if the text box has default input focus, and only disappears once the user starts typing.
  - The prompt text is restored if the text box is still empty when it loses input focus.

Recommended Sizing And Spacing

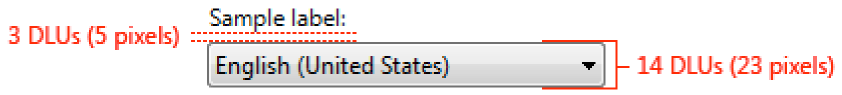


Figure 3-3: Recommended sizing and spacing for buttons

- **Choose a width appropriate for the longest valid data.** Drop-down lists cannot be scrolled horizontally, so users can see only what is visible in the control. (Note, however, that combo boxes can have AutoScroll functionality enabled.)
- **Include an additional 30%** (up to 200% for shorter text) for any text (but not numbers) that will be localized.
- **Choose a list length that eliminates unnecessary vertical scrolling.** Because drop-down lists are displayed on demand, their lists should show up to 30 items. Editable list boxes (those that don't have a drop-down button) should show between 3 and 12 items.

3.5.2 Labels

3.5.2.1 Control Labels

Write the label as a word or phrase, not as a sentence, and end it with a colon. Exceptions: Editable drop-down lists with prompts located where space is at a premium.

If a drop-down list or combo box is subordinate to a radio button or check box and is introduced by its label ending with a colon, don't put an additional label on the control.

Assign a unique access key for each label. For guidelines, see Keyboard.

Use sentence-style capitalization.

Position the label either to the left of or above the control, and align the label with the left edge of the control. If label is on the left, vertically align the label text with the control text.

- You may specify units (seconds, connections, and so on) in parentheses after the label.
- Don't make the content of the drop-down list or combo box (or its units label) part of a sentence, because this is not localizable.

3.5.3 Option Text

- Assign a unique name to each option.

- Use **sentence-style capitalization**, unless an item is a proper noun.
- Write the label as a word or phrase, not as a sentence, and use no ending punctuation.
- Use parallel phrasing, and try to keep the length about the same for all options.

3.5.4 Instructional Text

- If you need to add instructional text about a drop-down list or combo box, add it above the label. Use complete sentences with ending punctuation.
- Use **sentence-style capitalization**.
- Additional information that is helpful but not necessary should be kept short. Place this information either in parentheses between the label and colon, or without parentheses below the control.

3.5.5 Group Boxes

3.5.5.1 Is This The Right Control?

While group boxes are a strong visual means of indicating relationships, overusing them adds visual clutter and greatly reduces the space available on a surface. They are visually heavy and should be used sparingly—only when there isn't a better solution.

A design trend in Windows® is a simpler, cleaner appearance by eliminating unnecessary lines.

To decide whether a group box is necessary, consider these questions:

- **Is there more than one control in the group?** If not, use a plain text label instead. A rare exception is to use a group box with a single control to maintain consistency with other group boxes on the same surface.
- Are the controls related? Does showing the relationship add clarity? If not, present the controls separately outside of a group box.
- Are all the controls inside the group? If so, indicate the relationship on the larger surface, such as the parent dialog box or page.
- Can you effectively communicate the relationships using layout alone? If so, use layout instead. You can place related controls next to each other and put extra spacing between unrelated controls. You can also use headings and indenting to show hierarchical relationships.

- **Can you effectively communicate the relationships using a separator?** If so, use a separator instead. A separator is a horizontal line that unifies the controls below it. Separators provide a simpler, cleaner look. However, unlike group boxes, they work best when they span the full width of the surface. **Developers:** You can implement a separator with an etched rectangle with a height of one.
- **Can you effectively communicate the relationships without text?** If so, consider using graphic elements such as **backgrounds** or **aggregators**.

3.6 Guidelines (Page 94)

- **Don't nest group boxes.** Use layout to show relationships within a group box.
  - Don't put controls in group box labels. Exception: You can use a check box as a group box label if all of the controls inside the box are enabled and disabled by the check box.
- **Don't disable group boxes.** To indicate that a group of controls doesn't currently apply, disable all the controls within the group box, but not the group box itself. This approach is more accessible and can be supported consistently by all UI frameworks.

3.6.1 Labels

Label all group boxes.

- Don't assign an access key to the label. Doing so is unnecessary and makes the other access keys harder to assign. Instead, assign access keys to the controls within the group box. **Exception:** If a surface has many controls, there may not be enough access keys available. If so, reduce the number of access keys by assigning them to group boxes instead of the controls within the group boxes.
- Use **sentence-style capitalization**.
- Write the label using a noun or a noun phrase, not as a sentence, and use no ending punctuation, including colons.
- Use parallel phrasing for group box labels within the same surface.
- Keep group box labels concise. Don't use instructional text as the label. You can have instructional text within the group box, however.
- Don't repeat the group box label in control labels within the box. For example, if the group box is labeled *Alignment*, label the option buttons *Left*, *Right*, and so on, not *Left alignment* or *Right alignment*.
- Don't refer to group boxes in user interface text.

3.6.2 Using Links

3.6.2.1 Is This The Right Control?

To decide, consider these questions:

- Is the link used to navigate to another page, window, or Help topic; display a definition; initiate a command; or choose an option? If not, use another control.
- Would a command button be a better choice? Use a command button if: The control initiates an immediate action, including displaying a window, and that command relates to the primary purpose of the window.
  - A window is displayed to gather input or making choices, even if for a secondary command.
  - The label is short, consisting of four or fewer words, thus avoiding the awkward appearance of long buttons.
  - The command is not inline.
  - The control appears within a group of other related command buttons.
  - The action is destructive or irreversible. Because users associate links with navigation (and the ability to back out), links aren't appropriate for commands with significant consequences.
  - Similarly, in a **wizard** or **task flow**, the command represents commitment. In such windows, command buttons suggest commitment whereas links suggest navigating to the next step.

For a detailed comparison, see **Command Buttons vs. Links**.

3.6.2.2 Making Links Specific, Relevant, And Predictable

Concise links are more likely to be read than verbose links. **Eliminate unnecessary text and detail.** Link labels don't have to be comprehensive.

To evaluate your link text:

- Make sure the link text reflects the scenarios that the link supports.
- Make sure the results of the link are predictable. Users shouldn't be surprised by the results.

If you do only two things... (1) Make links discoverable by visual inspection alone. Users shouldn't have to interact with your program to find links. (2) Use links that give specific descriptive information about the result of clicking on the link, using as much text as necessary. Users should be able to accurately predict the result of a link from its link text and optional **infotip**.

3.6.3 Guidelines

3.6.3.1 Interaction

Display a busy pointer if the result of clicking a link isn't instantaneous. Without feedback, users might assume that the click didn't happen and click again.

3.6.4 Color

- **Use the theme or link system colors for visited and unvisited links.** The meaning of these colors is consistent across all programs. If for any reason users don't like these colors (perhaps for accessibility reasons), they can change them themselves.
- **For navigation links, use different colors for visited and unvisited links.** Keep the history of visited links only for the duration of the program instance. The visited color is important to indicate where users have already been, preventing them from unintentionally revisiting the same pages repeatedly.
- **For other types of links, don't use the visited link color.** There isn't sufficient value in identifying "visited" commands, for example.
- **Don't color 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. **Exception**: You can use colored text if all links are either underlined or placed within standard navigation or command locations.

3.6.5 Underlining

- **For links that are necessary to perform a primary task, provide visual clues so that users can recognize links by visual inspection alone.** These clues include underlining, graphics or bullets, and standard link locations. Users shouldn't have to hover over an object or attempt to click on it to determine if it is a link. Use underlined text if the link isn't obvious from its context.
- **Don't underline text that isn't a link because users may assume that it is a link.** Use italics where you'd otherwise use underlined text. Reserve underlining only for links.
- **When printing, don't print underlines or link colors.** Printed links have no value and are potentially confusing.

3.6.6 Text with Icon Links

- **Use the arrow icon only for command links.** Regular links shouldn't use the arrow icon unless they are being used as a substitute for command links in Windows XP.
- **Place the icon to the left of the text.** The icon needs to lead into the text visually.

- **Make the result of clicking the icon the same as clicking the text.** Doing otherwise would be unexpected and confusing.

3.6.7 Graphics-Only Links

Don’t use graphics-only links. Users have difficultly recognizing them as links and any text within the graphic (used to indicate their action when clicked) creates a localization problem.

3.6.8 Navigation Links

- **Make sure navigation links don’t require commitment.** Users should always be able to return to the initial state, either by using Back for inplace navigation or Cancel to close a new window.
- **Link to specific content rather than general content.** For example, it is better to link to the relevant section of a document than to link to the beginning.
- **Use a link only if the linked material is relevant, helpful, and not redundant.** Use restraint in navigation links—don’t use them just because you can.
- **If a link navigates to an external site, put the URL in the infotip** so that users can determine the target of the link.
- **Link only the first occurrence of the link text.** Redundant links are unnecessary and can make text difficult to read.
  - Exceptions: If an instruction has a link, put the link in the instruction.
- Link to later occurrences if they are far away from the first. For example, you can link redundantly in different sections within a Help topic.

3.6.9 Task Links

Use task links for commands that aren’t destructive or are easily reversible. Because users associate links with navigation (and the ability to back out), links aren’t appropriate for commands with significant consequences. Commands that display a dialog box or a confirmation are a good choice.

3.7 Menu Links (Page 102)

- **Group related navigation and task links into menus.** A menu of related links placed within a standard navigation or command location makes it easier to find and understand the links than when they’re placed separately.
- **For selection-dependent menus, remove menu links that don’t apply.** Don’t disable them. Doing so eliminates clutter and users won’t miss links that require selection.

- **For selection-independent menus, disable menu links that don't apply.** Don't remove them. Doing so makes the menus more stable and such links easier to find.

3.7.1 Link Infotips

If a link requires further explanation, **provide the explanation in either a supplemental explanation in a separate text control or an infotip**, but not both. Use complete sentences and ending punctuation. Providing both is unnecessary if the text is the same, and confusing if the text is different.

- Don't provide an infotip that is merely a restatement of the link text.

3.7.2 Text

- Don't assign an **access key**. Links are accessed using the Tab key.
- **Use links that give specific descriptive information about the result of clicking on the link**, using as much text as necessary. The link text should indicate the result of clicking on the link. **Users should be able to accurately predict the result of a link from its link text and optional infotip.**
  - For inline links: Preserve the capitalization and punctuation of the text.
  - Don't include ending punctuation in the link unless the text is a question.
  - Link on the most relevant part of the text and choose link text that is large enough to be easy to click.

Avoid putting two different inline links next to each other. Users are likely to believe they are a single link.

- For independent links (not inline): Use **sentence-style capitalization**.
- Don't use ending punctuation unless the link is a question.
- Use links that are clearly differentiated from the other links on the screen. Users should be able to accurately predict and differentiate between link targets.
- Don't add *Click* or *Click here* to the link text. It isn't necessary because a link implies clicking. Also, *Click here* and *here* alone convey no information about the link when read by a screen reader.

3.7.3 Navigation Links

- **Start the link with a noun and clearly describe where clicking the link will go.** Don't use ending punctuation. On occasion you may need to start navigation links with a verb, but don't use verbs that reiterate navigation that is already implied by the fact of linking, such as *View*, *Open*, or *Go to*.

- **Present a navigation link as a URL if it navigates to a Web page and you expect the target users to recall the URL and type it into a browser.** If possible, design such URLs to be short and easy to remember.
- **If the link includes a URL to a Web site starting with "www," omit the http:// protocol name and use lowercase text.**

3.7.4 Task Links

- **Start the link with an imperative verb and clearly describe the task that the link performs.** Don't use ending punctuation.
- **End the link with an ellipsis if the command needs additional information (including a confirmation) for successful completion.** Don't use an ellipsis when the successful completion of the task is to display another window—only when additional information is needed to perform the task.
- **If necessary, end a task link with "now" to distinguish it from a navigation link.**

3.7.5 Link Infotips

Use full sentences and ending punctuation.

For more guidelines and examples, see Tooltips and Infotips.

3.8 List Boxes (Page 105)

3.8.1 Is This The Right Control?

To decide, consider these questions:

- **Does the list present data, rather than program options?** Either way, a list box is a suitable choice regardless of the number of items. By contrast, **radio buttons** or **check boxes** are suitable only for a small number of program options.
- **Do users need to change views, group, sort by columns, or change column widths and order?** If so, use a **list view** instead.
- **Does the control need to be a drag source or a drop target?** If so, use a list view instead.
- **Do the list items need to be copied to or pasted from the clipboard?** If so, use a list view instead.

3.8.2 Single-Selection Lists

- Is the control used to choose one item from a list of mutually exclusive values? If not, use another control. For choosing multiple items, use a standard multiple-selection list, check box list, list builder, or add/remove list instead.

- Is there a default option that is recommended for most users in most situations? Is seeing the selected option far more important than seeing the alternatives? If so, consider using a drop-down list if you don't want to encourage users to make changes by hiding the alternatives.
- Does the list require constant interaction? If so, use a single-selection list to simplify the interaction.
- Does the setting seem like a relative quantity? Would users benefit from instant feedback on the effect of setting changes? If so, consider using a slider instead.
- Is there a significant hierarchical relationship between the list items? If so, use a tree view control instead.
- Is screen space at a premium? If so, use a drop-down list instead because the screen space used is fixed and independent of the number of list items.

3.8.3 Standard Multiple-Selection Lists And Check Box Lists

- Is multiple selection essential to the task or commonly used? If so, use a check box list to make multiple selection obvious, especially if your target users aren't advanced. Many users won't realize that a standard multiple-selection list supports multiple selection. Use a standard multiple-selection list if the check boxes would draw too much attention to multiple selection or result in too much screen clutter.
- Is the stability of the multiple selection important? If so, use a check box list, list builder, or add/remove list because clicking changes only a single item at a time. With a standard multiple selection list, it's very easy to clear all the selections—even by accident.
- Is the control used to choose zero or more items from a list of values? If not, use another control. For choosing one item, use a single-selection list instead.

3.8.4 Preview Lists

Are the options easier to select with images than with text alone? If so, use a preview list.

3.8.5 List Builders And Add/Remove Lists

- **Is the control used to choose zero or more items from a list of values?** If not, use another control. For choosing one item, use a single-selection list instead.
- **Does the order of the selected items matter?** If so, the **list builder** and **add/remove list** patterns support order, whereas the other multiple-selection patterns do not.
- **Is it important for users to see a summary of all the selected items?** If so, the list builder and add/remove list patterns display only the selected items, whereas the other multiple-selection patterns do not.

- **Are the possible choices unconstrained?** If so, use an add/remove list so that users can choose values not currently in the list.
- **Does adding a value to the list require a specialized dialog box for choosing objects?** If so, use an add/remove list and display the dialog box when users click Add.
- **Is screen space at a premium?** If so, use an add/remove list instead because it uses less screen space by not always showing the set of options.

3.8.6 Guidelines

3.8.6.1 Presentation

- **Sort list items in a logical order,** such as grouping related options together, placing most frequently used items first, or using alphabetical order. Sort names in alphabetical order, numbers in numeric order, and dates in chronological order. Lists with 12 or more items should be sorted alphabetically to make items easier to find.
  - Place options that represent All or None at the beginning of the list, regardless of sort order of the remaining items.
  - Enclose metaoptions in parentheses.
  - Don’t have blank list items—use metaoptions instead. Users don’t know how to interpret blank items, whereas the meaning of meta-options is explicit.

3.8.6.2 Interaction

- **Consider providing double-click behavior.** Double-clicking should have the same effect as selecting an item and performing its default command.
- **Make double-click behavior redundant.** There should always be a command button or context menu command that has the same effect.
- **If users can’t do anything with the selected items, don’t allow selection.**
  - When disabling a list box, also disable any associated labels and command buttons.
  - Don’t use the change of the selected item in a list box to:
    - Perform commands.
    - Display other windows, such as a dialog box to gather more input.
    - Dynamically display other controls related to the selected control (screen readers cannot detect such events). **Exception:** You can dynamically change static text used to describe the selected item.

- **Avoid horizontal scrolling.** Multicolumn lists rely on horizontal scrolling, which is generally harder to use than vertical scrolling. Multicolumn lists that require horizontal scrolling may be used when you have many alphabetically sorted items and sufficient screen space for a wide control.

3.9 Multiple-Selection Lists (Page 113)

- **Consider displaying the number of selected items below the list,** especially if users are likely to select several items. This information not only gives useful feedback, but it also clearly indicates that the list box supports multiple selection.
  - You can provide other selection metrics that might be more meaningful, such as the resources required for the selections.
- If there are potentially many list items and selecting or clearing all of them is likely, add Select all and Clear all command buttons.
- For standard multiple-selection lists, don't use multiple-selection mode because this selection mode has been deprecated. For equivalent behavior, use a check box list instead.

3.9.1 Default Values

- **Select the safest (to prevent loss of data or system access) and most secure option by default.** If safety and security aren't factors, select the most likely or convenient option. **Exception:** Don't select any items if the control represents a property in a **mixed state**, which happens when displaying a property for multiple objects that don't have the same setting.

3.9.1.1 Recommended Sizing And Spacing

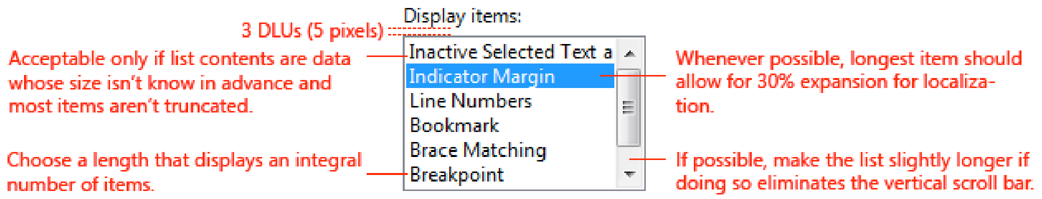

3 DLUs (5 pixels)
Display items:
Acceptable only if list contents are data whose size isn't know in advance and most items aren't truncated.
Choose a length that displays an integral number of items.
Inactive Selected Text a
Indicator Margin
Line Numbers
Bookmark
Brace Matching
Breakpoint
Whenever possible, longest item should allow for 30% expansion for localiza-tion.
If possible, make the list slightly longer if doing so eliminates the vertical scroll bar.

Figure 3-4: Recommended size and spacing for drop-down menus

- **Choose a list box width appropriate for the longest valid data.** Standard list boxes cannot be scrolled horizontally, so users can see only what is visible in the control.
- **Include an additional 30%** (up to 200% for shorter text) for any text (but not numbers) that will be localized.
- **Choose a list box height that displays an integral number of items.** Avoid truncating items vertically.

- **Choose a list box height that eliminates unnecessary vertical scrolling.** List boxes should display between 3 and 20 items without the need for scrolling. Consider making a list box slightly longer if doing so eliminates the vertical scroll bar. Lists with potentially many items should display at least five items to facilitate scrolling by showing more items at a time and making the scroll bar easier to position.
- **If users benefit from making the list box larger, make the list box and its parent window resizable.** Doing so allows users to adjust the list box size as needed. However, resizable list boxes should display no fewer than three items.

3.9.2 Labels

3.9.2.1 Control Labels

- All list boxes need labels. Write the label as a word or phrase, not as a sentence; use a colon at the end of the label.**Exception:** Omit the label if it is merely a restatement of a dialog box's **main instruction**. In this case, the main instruction takes the colon (unless it's a question) and access key.
- If a list box is subordinate to a radio button or check box and is introduced by that control's label ending with a colon, don't put an additional label on the list box control.
- Assign a unique **access key**. For guidelines, see **Keyboard**.
- Use **sentence-style capitalization**.
- Position the label either to the left of or above the control, and align the label with the left edge of the control. If label is on the left, vertically align the label text with the first line of text in the control.
- For multiple-selection list boxes, use a label that clearly indicates multiple selection is possible. Check box list labels can be less explicit.

3.9.3 Option Text

- Assign a unique name to each option.
- Use **sentence-style capitalization**, unless an item is a proper noun.
- Write the label as a word or phrase, not as a sentence, and use no ending punctuation.
- Use parallel phrasing, and try to keep the length about the same for all options.

3.9.4 Instructional And Supplemental Text

- If you need to add instructional text about a list box, add it above the label. Use complete sentences with ending punctuation.

- Use **sentence-style capitalization**.
- Additional information that is helpful but not necessary should be kept short. Place this text either in parentheses between the label and colon, or without parentheses below the control.

3.10 List Views (Page 17)

3.10.1 Is This The Right Control?

A list view is more than just a more flexible and functional list box: its extra functionality results in different usage. The following table shows the comparison.

To decide if this is the right control, consider these questions:

- Does the list present data, rather than program options? If not, consider using a list box instead.
- Do users need to change views, group, sort by columns, or change column widths and order? If not, use a list box instead.
- Does the control need to be a drag source or a drop target? If so, use a list view.
- Do the list items need to be copied to or pasted from the clipboard? If so, use a list view.

3.10.2 Check Box List Views

- Is the control used to choose zero or more items from a list of data? To choose one item, use single selection instead.
- Is multiple selection essential to the task or commonly used? If so, use a check box list view to make multiple selection obvious, especially if your target users aren't advanced. If not, use a standard multiple-selection list view if the check boxes would draw too much attention to multiple selection or result in too much screen clutter.
- Is the stability of the multiple selection important? If so, use a check box list, list builder, or add/remove list because clicking changes only a single item at a time. With a standard multiple selection list, it's very easy to clear all the selections—even by accident.

3.10.3 Guidelines

3.10.3.1 Presentation

- **Sort list items in a logical order.** Sort names in alphabetical order, numbers in numeric order, and dates in chronological order.

- **If appropriate, allow users to change the sort order.** User sorting is important if the list has many items or if there are scenarios where items are found more effectively using a sort order other than the default.
- **Use the Always Show Selection attribute** so that users can readily determine the selected item, even when the control doesn't have focus.
  - **Avoid presenting empty list views.** If users create a list, initialize the list with instructions or example items that users might need.
  - **If users can change views, group, sort by columns, or change columns and their widths and order, make those settings persist so they take effect the next time the list view is displayed.** Make them persist on a per-list view, per-user basis.

3.10.4 Interaction

- **Use single-click to select the list item the user is pointing to. Exception:** For the command link list pattern, single-click selects the item and either closes the window or navigates to the next page.
- **Consider providing double-click behavior.** Double clicking should have the same effect as selecting an item and performing its default command.
- **Make double-click behavior redundant.** There should always be a command button or context menu command that has the same effect.
- If a list item requires further explanation, **provide the explanation in an infotip**. Use complete sentences and ending punctuation.
- **Provide context menus of relevant commands.** Such commands include Cut, Copy, Paste, Remove or Delete, Rename, and Properties.
- **If users can change the sort order and grouping, provide Sort By and Group By context menus.** The first click on a column name sorts or groups the list in the ascending order for that column, the second click sorts or groups in descending order. Use the previous order (from another column) as the secondary key.
- **Make the list view column header accessible using the keyboard. Developers:** You can do this by setting focus on the column header control. This capability is new to Windows Vista®.
- **When disabling a list view, also disable any associated labels and command buttons.**
- **Avoid horizontal scrolling.** The List mode uses horizontal scrolling. This mode is usually the most compact, but horizontal scrolling is generally harder to use than vertical scrolling. Consider using the Small Icon view instead if compactness isn't important. However, List mode is a good choice when there are many alphabetically sorted items and sufficient screen space for a wide control.

3.10.5 Multiple-Selection Lists

- **Consider displaying the number of selected items below the list**, especially if users are likely to select several items. This information not only gives useful feedback, but it also clearly indicates that the list view supports multiple selection.
  - For check box list views, if there are potentially many items and selecting or clearing all of them is likely, add Select all and Clear all command buttons.
  - **Use mixed-state check boxes to indicate partial selection of the items in a container.** The mixed state is not used as a third state for an individual item.

3.10.6 Changing Views

If users can change views:

- **Choose the most convenient view by default.** Any changes users make should be made persistent on a per-list view, per-user basis.
- **Change the view using a split button, menu button, or drop-down list.** Whenever practical, use a split button on the toolbar and change the button label to reflect the current view.
- **Choose default column widths appropriate for the longest data.** List views automatically truncate long data with ellipses, so the column widths are appropriate if few ellipses are displayed by default. While users can resize columns, prefer other solutions:

1. Size each column width to fit its data.
2. Size the control width to fit its columns plus any likely scrollbars.
3. If necessary, use horizontal scrolling.
4. Have truncated data only for odd-sized items or as a last resort.
5. If normal-sized data must be truncated by default, make the window and list view resizable. Include an additional 30% (up to 200% for shorter text) for any text (but not numbers) that will be localized.

- Choose an appropriate default column order. Generally, order the columns as follows:

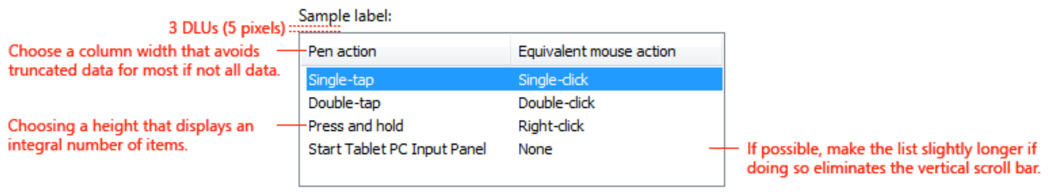
1. First, the item name or identifying data.
2. Next, other data useful in differentiating the list items.
3. Next, the most useful (preferably short or fixed length) data.
4. Next, less useful (preferable short or fixed length) data.
5. Last, long, variable-length data.

Long, variable length-data is placed in the last columns to reduce the need for horizontal scrolling. Within these categories, place related information together in a logical sequence.

When appropriate, allow users to add and remove columns, as well as change the order. Display the most useful columns by default. This is achieved with the Header Drag Drop attribute.

- Choose an alignment appropriate for the data. Use the following rules: Right-align numbers, currencies, and times.
- Left-align text, IDs (even if numeric), and dates.
- For sortable column headings, **the first click on a heading sorts the list in ascending order for the column, the second click sorts in descending order.**
  - Use the previous sort order (from another column) as the secondary sort key.
  - **Use the Full Row Select attribute** so that users can readily determine the selected items in all columns.
  - **Don't use a sortable column header unless the data can be sorted.**
  - **Don't use a column header if there is only one column and there is no need to reverse sort.** Use a label instead to identify the data.

3.11 Recommended Sizing And Spacing (Page 127)


3 DLUs (5 pixels)
Sample label:
Choose a column width that avoids truncated data for most if not all data.
Choosing a height that displays an integral number of items.
If possible, make the list slightly longer if doing so eliminates the vertical scroll bar.

Pen action	Equivalent mouse action
Single-tap	Single-click
Double-tap	Double-click
Press and hold	Right-click
Start Tablet PC Input Panel	None

Figure 3-5: Recommended sizing and spacing of sample label

- Choose a list view height that displays an integral number of items. Avoid truncating items vertically.
- Choose a list view size that eliminates unnecessary vertical and horizontal scrolling in all supported views. List views should display between 3 and 20 items. Consider making a list view slightly larger if doing so eliminates a scroll bar. Lists with potentially many items should display at least five items to facilitate scrolling by showing more items at a time and making the scroll bar easier to position.
- If users benefit from making the list view larger, make the list view and its parent window resizable. Doing so allows users to adjust the list view size as needed. However, resizable list views should display no fewer than three items.

3.11.1 Labels

3.11.1.1 Control Labels

- All list views need labels. Write the label as a word or phrase, not as a sentence, ending with a colon using static text.
- Assign a unique **access key**. See **Keyboard** for guidelines on assigning access keys.
- Use **sentence-style capitalization**.
- Position the label above the control and align the label with the left edge of the control.
- For multiple-selection list views, write the label that clearly indicates multiple selection is possible. Check box list view labels can be less explicit.
- You may specify units (seconds, connections, and so on) in parentheses after the label.

3.11.1.2 Heading Labels

- Keep the heading labels brief (three words or fewer).
- Use a single noun or noun phrase with no ending punctuation.
- Use **sentence-style capitalization**.
- Align the heading the same way as the data.

3.11.1.3 Group Labels

- Use the following group labels for high-level collections: Names: Use first letter of name or letter ranges.
- Sizes: Unspecified, 0 KB, 0–10 KB, 10–100 KB, 100 KB–1 MB, 1–16 MB, 16–128 MB
- Dates: Today, Yesterday, Last week, Earlier this year, and A long time ago.
- Otherwise, group labels use the exact text of the data being grouped, including capitalization and punctuation.

3.11.2 Data Text

- Use sentence-style capitalization.
- If you need to add instructional text about a list view, add it above the label. Use complete sentences with ending punctuation.
- Use sentence-style capitalization.

- Additional information that is helpful but not necessary should be kept short. Place this information either in parentheses between the label and colon or without parentheses below the control.

3.11.3 Progress Bars

3.11.3.1 Is This The Right Control?

To decide, consider these questions:

- **Will the operation complete in about five seconds or less?** If so, use an **activity indicator** instead, because displaying a progress bar for such a short duration would be distracting. If the operation usually takes five seconds or less but sometimes takes more, start with a busy pointer and convert to a progress bar after five seconds.
- **Is an indeterminate progress bar used to wait for the user to complete a task?** If so, don't use a progress bar. Progress bars are for computer progress, not user progress.
- **Is an indeterminate progress bar combined with an animation?** If so, use just the animation instead. The indeterminate progress bar is effectively a generic animation and adds no value to the animation.
- **Is the operation a very lengthy (longer than two minutes) background task for which users are more interested in completion than progress?** If so, use a **notification** instead. In this case, users do other tasks in the meantime and are not monitoring the progress. Using a notification allows users to perform other tasks without disruption. Examples of such lengthy operations include printing, backup, system scans, and bulk data transfers or conversions.
- **When the operation is complete, will users be able to replay the results?** If so, use a slider instead. Examples of such operations include video and audio recording and playback.

3.11.4 Guidelines

3.11.4.1 General

- Provide progress feedback when performing a lengthy operation. Users should never have to guess if progress is being made.
- Clearly indicate real progress. The progress bar must advance if progress is being made. If the range of expected completion times is large, consider using a non-linear scale to indicate progress for the longer times. You don't want users to conclude that your program has locked up when it hasn't.

- Clearly indicate lack of progress. The progress bar must not advance if no progress is being made. You don’t want users to wait indefinitely for an operation that is never going to complete.
- 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 provide unnecessary details.** Generally users don’t care about the details of the operation being performed. For example, users of a setup program don’t care about the specific file being copied or that system components are being registered because they have no expectations about these details. Typically, a well-labeled progress bar alone provides sufficient information, so provide additional progress information only if users can do something with it. Providing details that users don’t care about makes the user experience overly complicated and technical. If you need more detailed information for debugging, don’t display it in release builds.
  - **Provide useful animations.** If done well, animations improve the user experience by helping users visualize the operation. Good animations have more impact than text alone. For example, the progress bar for the Outlook Delete command displays the Recycle Bin for the destination if the files can be recovered, but no Recycle Bin if the files can’t be recovered.
  - **Don’t use unnecessary animations.** Animations can be misleading because they usually run in a separate thread from the actual task and therefore can suggest progress even if the operation has locked up. Also, if the operation is slower than expected, users sometimes assume that the animation is part of the reason. Consequently, only use animations when there is a clear justification; don’t use them to try to entertain users.
  - **Position animations centered over the progress bar.** Put the animation above the progress bar labels, if you have any. If there is a Cancel or Stop button to the right of the progress bar, include the button when determining the center.
  - **Play a sound effect at the completion of an operation only if it is very lengthy (longer than two minutes), infrequent, and important.** If the user is likely to walk away from an important operation while it is processing, a sound effect restores the user’s attention. Using a sound effect upon completion in other circumstances would be a distracting annoyance.
  - **Don’t steal input focus to show a progress update or completion.** Users often switch to other programs while waiting and don’t want to be interrupted. Background tasks must stay in the background.

- **Don’t worry about technical support.** Because the feedback provided by progress bars isn’t necessarily accurate and is fleeting, progress bars aren’t a good mechanism for providing information for technical support. Consequently, if the operation can fail (as with a setup program), don’t provide additional progress information that is only useful to technical support. Instead, provide an alternative mechanism such as a log file to record technical support information.
- **Don’t put the percentage complete or any other text on a progress bar.** Such text isn’t accessible and isn’t compatible with using themes.
- **Don’t combine a progress bar with a busy pointer.** Use one or the other, but not both at the same time.
- **Don’t use vertical progress bars.** Horizontal progress bars have a more natural mapping and better flow.

3.11.5 Determinate 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.
- **Clearly indicate the progress phase.** The progress bar must be able to indicate if the operation is in the beginning, middle, or end of an operation. For example, progress bars that immediately shoot to 99% completion, then stay there for a long time are particularly uninformative and annoying. In these cases, the progress bar should be set initially to at most 33% to indicate that the operation is still in the beginning phase.
- **Clearly indicate completion.** Don’t let a progress bar go to 100% unless the operation has completed.
- **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.
- **Don’t back up progress.** As with a restart, a progress bar loses its value if it backs up. Always increase progress monotonically. However, you can have a time remaining estimate that increases (as well as decreases) because the rate of progress may vary.

3.11.6 Indeterminate Progress Bars

- **Use indeterminate progress bars only for operations whose overall progress cannot be determined.** Use indeterminate progress bars for operations that require an unbounded amount of time or that access an unknown number of objects. Use timeouts to give bounds to time-based operations.
- **Convert to a determinate progress bar once the overall progress can be determined.** For example, if it takes significantly longer than two seconds to determine the number of objects, you can use an indeterminate progress bar while the objects are counted, and then convert to a determinate progress bar.
- **Don’t combine indeterminate progress bars with percent complete or time remaining estimates.** If you can provide this information, use a determinate progress bar instead.
- **Don’t combine indeterminate progress bars with animations.** An indeterminate progress bar is effectively a generic animation, so you should use one or the other but never both

3.11.7 Modeless Progress Bars

- If users can do something productive while the operation is in progress, provide modeless feedback. You might need to disable a subset of functionality that requires the operation to complete.
- If the window has an address bar, display the modeless progress in the address bar. Otherwise, if the window has a status bar, display the modeless progress in the status bar. Put any corresponding text to its left in the status bar.

3.12 Modal Progress Bars (Page 137)

- Place modal progress bars on progress pages or progress dialog boxes.
- Provide a command button to halt the operation if it takes more than a few seconds to complete, or has the potential never to complete. Label the button Cancel if canceling returns the environment to its previous state (leaving no side effects); otherwise label the button Stop to indicate that it leaves the partially completed operation intact. You can change the button label from Cancel to Stop in the middle of the operation if at some point it isn’t possible to return the environment to its previous state. Center the command button vertically with the progress bar instead of aligning their tops.

3.12.1 Time Remaining

- **Use the following time formats.** Start with the first of the following formats where the largest time unit isn’t zero, and then change to the next format once the largest time unit becomes zero.
- **For progress bars:**

  - **If related information is shown in a colon format:** Time remaining: h hours, m minutes Time remaining: m minutes, s seconds Time remaining: s seconds.
  - **If screen space is at a premium:** h hrs, m mins remaining m mins, s secs remaining s seconds remaining.
  - **Otherwise:** h hours, m minutes remaining m minutes, s seconds remaining s seconds remaining**For title bars:** hh:mm remaining mm:ss remaining 0:ss remainingThis compact format shows the most important information first so that it isn't truncated on the taskbar.
- **Make estimates accurate, but don't give false precision.** If largest unit is hours, give minutes (if meaningful) but not seconds.
- **Keep the estimate up-to-date.** Update time remaining estimates at least every 5 seconds.
- **Focus on the time remaining** because that is the information users care about most. Give total elapsed time only when there are scenarios where elapsed time is helpful (such as when the task is likely to be repeated). If the time remaining estimate is associated with a progress bar, don't have percent complete text because that information is conveyed by the progress bar itself.
- **Be grammatically correct.** Use singular units when the number is one.
- **Use sentence-style capitalization.**

3.12.2 Progress Bar Colors

- **Use red or yellow progress bars only to indicate the progress status, not the final results of a task.** A red or yellow progress bar indicates that users need to take some action to complete the task. If the condition isn't recoverable, leave the progress bar green and display an error message.
- **Turn the progress bar red when there is a *user recoverable* condition that prevents making further progress.** Display a message to explain the problem and recommend a solution.
- **Turn the progress bar yellow to indicate either that the user has paused the task or that there is a condition that is impeding progress** but progress is still taking place (as, for example, with poor network connectivity). If the user has paused, change the Pause button label to Resume. If progress is impeded, display a message to explain the problem and recommend a solution.

3.12.3 Meters

- **Use progress bars only for progress.** Use meters to indicate percentages that aren't related to progress.

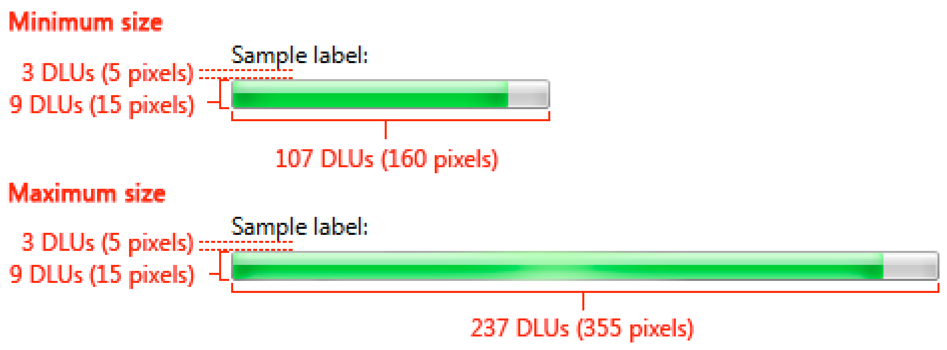


Figure 3-6: Progress bars

- Always use the recommended progress bar height. **Exception:** You may use a different height if the parent window doesn't support the recommended height.
- Use the minimum width if you want to make the progress bar unobtrusive.
- Don't use widths longer than the maximum recommended. The progress bar doesn't have to fill the available space.
- Center the progress bar horizontally if the window is much wider than the maximum recommended width.

3.12.4 Labels

3.12.4.1 Progress Bar Labels

- **Use a concise label with a static text control to indicate what the operation is doing.** Start the label with a verb (for example, *Copying*) and end with an ellipsis. This label may change dynamically if the operation has multiple steps or is processing multiple objects.
- Don't assign a unique **access key** because the control isn't interactive.
- Use **sentence-style capitalization**.
- If the operation was not directly initiated by the user, you can include an additional label to give the context and apologize for the interruption. Start this extra label with the phrase "Please wait while". This label should not change during the operation.
- Position the label above the progress bar and align the label with the left edge of the progress bar.

3.12.4.2 Progress Bar Details

- Provide details in static text, preceding the data with a label ending with a colon. Specify units (seconds, kilobytes, and so on) after the details text.
- Use sentence-style capitalization.

- Position the details below the progress bar and align the label with the left edge of the progress bar.
- Don't give the percentage completed or remaining because that information is conveyed by the progress bar itself.

3.12.5 Cancel Button

- Label the button Cancel if canceling returns the environment to its previous state (leaving no side effect); otherwise, label the button Stop to indicate that it leaves the partially completed operation intact.
- You can change the button label from Cancel to Stop in the middle of the operation if at some point it isn't possible to return the environment to its previous state.

3.12.6 Progress Dialog Box Titles

- If the progress bar is displayed in a modal dialog box, the dialog box title should be the name of the program or the name of the operation. Don't use what should be the progress bar label for the dialog box title.
- If the progress bar is displayed in a modeless dialog box, optimize the title for display on the taskbar by concisely placing the distinguishing information first. Example: "66% Complete."

3.13 Progressive Disclosure Controls (Page 142)

3.13.1.1 Is This The Right Control?

To decide, consider these questions:

- **Do users need to see the information in some but not all scenarios, or some but not all of the time?** If so, displaying the information using progressive disclosure simplifies the baseline experience, yet allows users to access the information easily.
  - **If the information is displayed by default, are users ever likely to choose to hide it?** Are there scenarios where users will need more space? Are users sufficiently motivated to customize the user interface (UI)? If not, display the information without using progressive disclosure.
  - **Is the additional information advanced, substantial, complex, or related to an independent subtask?** If so, consider displaying the information in a separate window using **command buttons** or **links** instead of using a progressive disclosure control. (Additional information is advanced if it is intended for advanced users. It's complex if it makes other information hard to read or lay out.)

– **Is the additional information a sentence or sentence fragment that describes what an item does or how it can be used?** If so, consider using a **tooltip** or **infotip**.
– **Is the additional information related to the current task, but independent of the currently displayed information?** If so, consider using **tabs** instead. However, collapsible lists are often preferable to tabs because they are more flexible and scalable.
– **Is showing or hiding the additional information essentially a data filter?** If so, consider using a **drop-down list** or **check boxes** instead to apply the filter to the entire list.

3.13.2 Guidelines

3.13.2.1 General

- Select the progressive disclosure pattern based on its usage.
- Don't use links for progressive disclosure controls. Use only the progressive disclosure controls presented in the Usage patterns section. However, *do* use links to navigate to Help topics.

3.13.3 Interaction

- For chevrons and arrows that aren't directly labeled, use tooltips to describe what they do.

  – **If a user expands or collapses an item, make the state persist so it takes effect the next time the window is displayed**, unless users are likely to prefer starting in the default state. Make the state persist on a per-window, per-user basis.
  – **Make sure that all expanded content can be collapsed and vice versa, and that the inverse operation is obvious.** Doing so encourages exploration and reduces frustration. The best way to make the inverse operation obvious is to keep the control in the same fixed location. If you need to move the control, keep it in the same relative location within a visually distinct area.
  – **Use only the access keys appropriate for the progressive disclosure pattern**, as listed in the Usage patterns section. Don't use Enter to activate progressive disclosure.

3.13.4 Presentation

- Don't use triangular-shaped arrowheads for a purpose other than progressive disclosure.
- Remove (don't disable) progressive disclosure controls that don't apply in the current context. Progressive disclosure controls should always deliver on their promise, so remove them when there isn't more information to give.

3.13.5 Chevrons

- Use single chevrons to show or hide in place. Use double chevrons to show or hide using a pop-up menu. You should always use double chevrons for command buttons with internal labels, however.
- Provide a visual relationship between the chevron and its associated control. Because in-place chevrons are placed to the right of their associated UI and right aligned, there can be quite a distance between a chevron and its associated control.

3.13.6 Arrows

- **Don't use arrow graphics that could be confused with Back, Forward, Go, or Play.** Use simple triangular-shaped arrowheads (arrows without stems) on neutral backgrounds

3.13.6.1 Recommended Sizing And Spacing

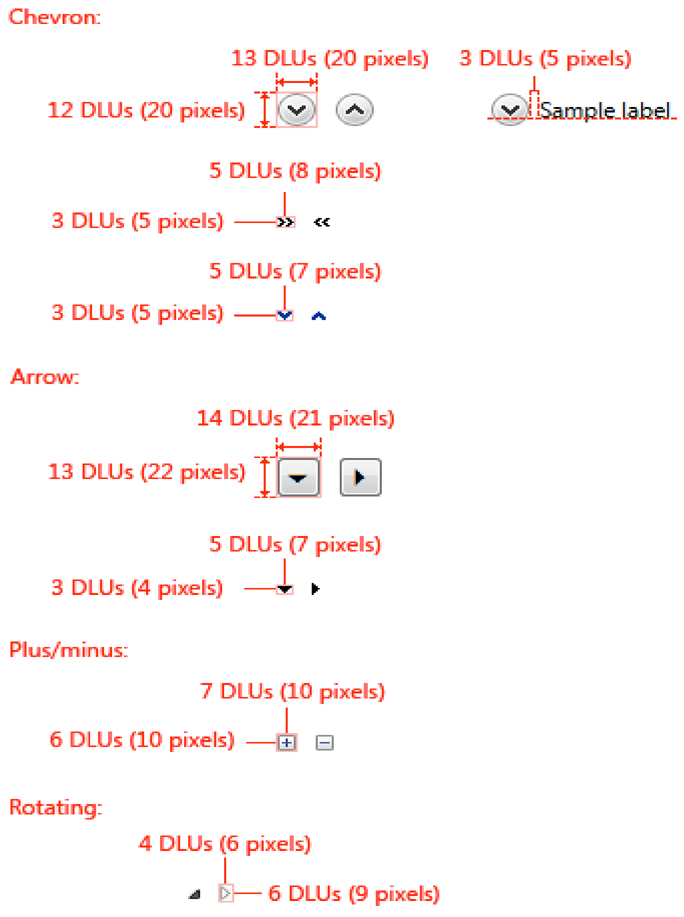


Figure 3-7: Using arrows

3.13.7 Labels

- For chevrons on a command button with an external label: Assign a unique **access key**. For assignment guidelines, see **Keyboard**.
- Use **sentence-style capitalization**.
- Write the label as a phrase and use no ending punctuation.
- Write the label so that it describes the effect of clicking the button, and change the label when the state changes.
- 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 <object name>.** 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 <object name>.** Use for other object types, such as folders.
- For chevrons on a command button with an internal label, follow the standard **command button label** guidelines.

3.13.8 Radio Buttons

3.13.8.1 Is This The Right Control?

To decide, consider these questions:

- Is the control used to choose one option from a set of mutually exclusive choices? If not, use another control. To choose multiple options, use check boxes, a multiple-selection list or a check box list instead.
- Is the number of options between two and seven? Since the screen space used is proportional to the number of options, keep the number of options in a group between two and seven. For eight or more options, use a drop-down list or single-selection list.

- Would a check box be a better choice? If there are only two options, you could use a single check box instead. However, check boxes are suitable only for turning a single option on or off, whereas radio buttons can be used for completely different alternatives. If both solutions are possible:
  - Use radio buttons if the meaning of the cleared check box isn't completely obvious.
  - Use radio buttons on wizard pages to make the alternatives clear, even if a check box is otherwise acceptable.
  - Use radio buttons if you have enough screen space and the options are important enough to be a good use of that screen space. Otherwise, use a check box or drop-down list.
  - Use a check box if there other check boxes on the page.
- **Would a drop-down list be a better choice?** If the default option is recommended for most users in most situations, radio buttons might draw more attention to the options than necessary. Consider using a drop-down list if you don't want to draw attention to the options, or you don't want to encourage users to make changes. A drop-down list focuses on the current selection, whereas radio buttons emphasize all options equally.
- **Would a set of command buttons, command links, or a split button be a better choice?** If the radio buttons are used only to affect how a command is performed, it is often better to present the command variations instead. Doing so allows users to choose the right command with a single interaction.
- **Do the options present program options, rather than data?** The options' values shouldn't be based on context or other data. For data, use a drop-down list or single-selection list.
- If the control is used on a wizard page or control panel, **is the control a response to the main instruction and can users later change the choice?** If so, consider using command links instead of radio buttons to make the interaction more efficient.
- **Are the values non-numeric?** For numeric data, use **text boxes**, **drop-down lists**, or **sliders**.

3.13.9 Guidelines

3.13.9.1 General

- **List the options in a logical order,** such as most likely to be selected to least, simplest operation to most complex, or least risk to most. Alphabetical ordering is not recommended because it is language dependent and therefore not localizable.
- **If none of the options is a valid choice, add another option to reflect this choice,** such as *None* or *Does not apply*.

- **Prefer to align radio buttons vertically instead of horizontally.** Horizontal alignment is harder to read and localize.
- **Reconsider using group boxes to organize groups of radio buttons—**this often results in unnecessary screen clutter.
- **Don't use radio button labels as group box labels.**
- Don't use the selection of a radio button to:
  - Perform commands.
  - Display other windows, such as a dialog box to gather more input.
  - Dynamically show or hide other controls related to the selected control (screen readers cannot detect such events). However, you can change text dynamically based on the selection.

3.13.10 Subordinate Controls

- Place subordinate controls to the right of or below (indented, flush with the radio button label) the radio button and its label. End the radio button label with a colon.
  - **Leave dependent editable text boxes and drop-down lists enabled if they share the radio button's label.** When users type or paste anything into the box, select the corresponding option automatically. Doing so simplifies the interaction.
  - **Avoid nesting radio buttons with other radio buttons or check boxes.** If possible, keep all the options at the same level.
- If you do nest radio buttons with other radio buttons or check boxes, **disable these subordinate controls until the high-level option is selected.** Doing so avoids confusion about the meaning of the subordinate controls.

3.13.11 Default Values

Because a group of radio buttons represents a set of mutually exclusive choices, always have one radio button selected by default. Select the safest (to prevent loss of data or system access) and most secure and private option. If safety and security aren't factors, select the most likely or convenient option. Exceptions: Don't have a default selection if:

- There is no acceptable default option for safety, security, or legal reasons and therefore the user must make an explicit choice.

If the user doesn't make a selection, display an error message to force one.

The user interface (UI) must reflect the current state and the option hasn't been set yet. A default value would incorrectly imply that the user doesn't need to make a selection.

The goal is to collect unbiased data. Default values would bias data collection.

The group of radio buttons represents a property in a mixed state, which happens when displaying a property for multiple objects that don't have the same setting. Don't display an error message in this case since each object has a valid state.

Make the first option the default option, since users often expect that—unless that order isn't logical. To do this, you might need to change the option labels.

3.13.12 Recommended Sizing And Spacing

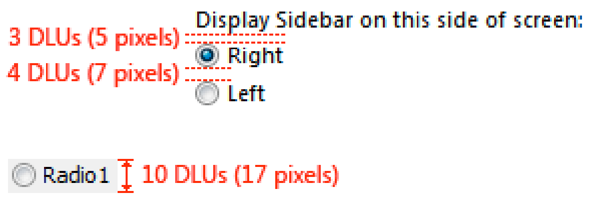


Figure 3-8: Recommended sizing and spacing of radio buttons

3.13.13 Labels

3.13.13.1 Radio Button Labels

- Label every radio button.
- Assign a unique **access key** to each label. For assignment guidelines, see **Keyboard**.
  - Use sentence-style capitalization.
    - Write the label as a phrase, not as a sentence, and use no ending punctuation. **Exception:** If a radio button label also labels a subordinate control that follows it, end the label with a colon.
    - Use parallel phrasing, and try to keep the length about the same for all labels.
    - Focus the label text on the differences among the options. If all the options have the same introductory text, move that text to the group label.
    - Use positive phrasing. For example, use *do* instead of *do not*, and *print* instead of *do not print*.
    - Describe just the option with the label. Keep labels brief so it's easy to refer to them in messages and documentation. If the option requires further explanation, provide the explanation in a **static text** control using complete sentences and ending punctuation.

Note: Adding an explanation to one radio button doesn't mean that you have to provide explanations for all the radio buttons. Provide the relevant information in the label if you can, and use explanations only when necessary. Don't merely restate the label for consistency.

- **If an option is strongly recommended, add "(recommended)" to the label.** Be sure to add to the control label, not the supplemental notes.
- **If an option is intended only for advanced users, add "(advanced)" to the label.** Be sure to add to the control label, not the supplemental notes.
- If you must use multiline labels, align the top of the label with the radio button.
- Don't use a subordinate control, the values it contains, or its units label to create a sentence or phrase. Such a design isn't localizable because sentence structure varies with language.

3.13.13.2 Radio Button Group Labels

- Use the group label to explain the purpose of the group, not how to make the selection. Assume that users know how to use radio buttons. For example, don't say "Select one of the following choices".
- All radio button groups need labels. Write the label as a word or phrase, not as a sentence, ending with a colon using static text or a group box. **Exception:** Omit the label if it is merely a restatement of a dialog box's **main instruction**. In this case, the main instruction takes the colon (unless it's a question) and access key (if there is one)

3.13.14 Search Boxes

3.13.14.1 Guidelines

3.13.14.1.1 Location

- For application windows, locate Search in the upper-right corner.
- For popup windows, locate Search wherever is most sensible and convenient.

  Exception: If Search is usually the first thing users do in a window (the primary entry point), center it at the top of the window.

3.13.14.1.2 Look

Use the standard search button graphics. There are three versions:

- **Magnifying glass search symbol only (no button on hover).** Use for Instant search.

- **Magnifying glass search symbol with button.** Use when button needs to be clicked to start the search.
- **Magnifying glass search symbol with button and drop-down arrow.** Add a drop-down arrow when users can change the scope or when other settings are available. For Instant search, use a drop-down arrow only, and show a button on hover.

For regular search, show the drop-down arrow on a button.

- Don't use a label; use an optional prompt instead. If users tend to assume that the search is a generic file search, use the prompt to give the scope. Otherwise, use *Type to search* or a similar, concise phrase.

3.13.14.1.3 Interaction

- **On input focus, automatically select any previously entered text.** Doing so allows users to enter a new search by typing, or to modify the previous search by positioning the caret using the arrow keys.
  - Assign the keyboard shortcut for the Search box to be Ctrl-E. For more information about keyboard shortcut assignments, see Windows Keyboard Shortcut Keys.

3.13.14.1.4 Functionality

- **Support Instant search whenever possible.** Provide both regular and Instant searches if there are scenarios where regular searching is worth the extra wait time.
- Regular searches must return relevant results within five seconds and Instant search must return results within two seconds. After this point, Search may continue to fill in less relevant results over time as long as the program is responsive and users can perform other tasks. You may have to index your search data to ensure this responsiveness.
- If you provide both regular and Instant search modes, the Instant search results must be a subset of the regular search results.
- All searching is prefix-based (no substring or suffix searching). The use of trailing wildcard characters is optional and doesn't affect the results. If multiple words are entered, use OR searching.
- A successful search adds a virtual page with the search results to the Back stack and Address bar. Multiple searches result in a single virtual page, so clicking Back always returns the original page.
- If necessary for scale, rank the search results by relevance.
- A blank search returns the original page.

3.13.14.2 Recommended Sizing And Spacing

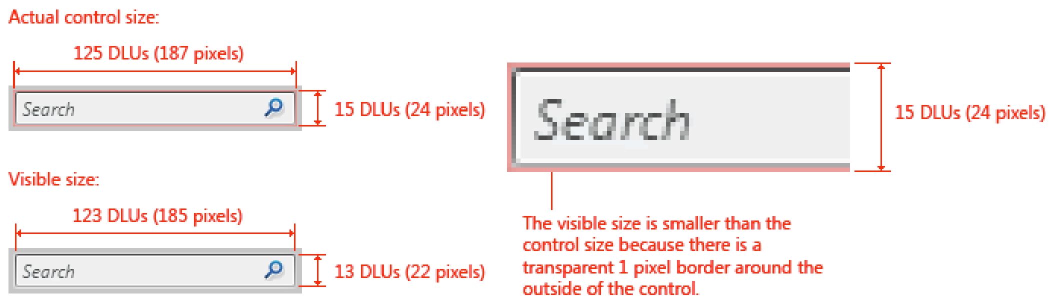


Figure 3-9: Recommended sizing and spacing of search fields

3.13.15 Text

- For the wording of the prompt in the Search box, either make it an instruction (for example, *Type to search*) or indicate the scope of the search (for example, *Search for pictures*).
- Prompt text should be brief. A single word or short phrase should suffice.
- Use **sentence-style capitalization**.
- Don't use ending punctuation or ellipsis.

3.13.16 Sliders

3.13.16.1 Guidelines

- **Use a natural orientation.** For example, if the slider represents a real-world value that is normally shown vertically (such as temperature), use a vertical orientation.
- **Orient the slider to reflect the culture of your users.** For example, Western cultures read from left to right, so for horizontal sliders, put the low end of the range on the left and the high end on the right. For cultures that read from right to left, do the opposite.
- **Size the control so that a user can easily set the desired value.** For settings with discrete values, make sure the user can easily select any value using the mouse.
- **Consider using a nonlinear scale if the range of values is large and users will likely select values at one end of the range.** For example, time value might be 1 minute, 1 hour, 1 day, or 1 month.
- **Whenever practical, give immediate feedback while or after a user makes a selection.** For example, the Microsoft® Windows® volume control beeps to indicate the resulting audio volume.

- **Use labels to show the range of values. Exception:** If the slider is vertically oriented and the top label is Maximum, High, More, or equivalent, you can omit the other labels since the meaning is clear.
- Use tick marks when users need to know the approximate value of the setting.
- Use tick marks and a value label when users need to know the exact value of the setting they choose. Always use a value label if a user needs to know the units to make sense of the setting.
- For horizontally-oriented sliders, place tick marks under the slider. For vertically-oriented sliders, place tick marks to the right for Western cultures; for cultures that read from right to left, do the opposite.
- Place the value label completely under the slider control so that the relationship is clear. Incorrect:
- When disabling a slider, also disable any associated labels.
- Don’t use both a slider and a numeric text box for the same setting. Use only the more appropriate control. Exception: Use both controls when the user needs both immediate feedback and the ability to set an exact numeric value.
- Don’t use a slider as a progress indicator.
- Don’t change the size of the slider indicator from the default size.
- Don’t label every tick mark.

3.13.16.2 Recommended Sizing And Spacing

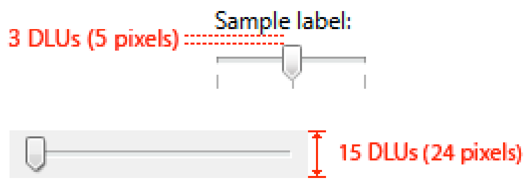


Figure 3-10: Recommended sizing and spacing of slider bars

3.13.17 Labels

3.13.17.1 Slider Labels

- Use a **static text** label ending with a colon, or a group box label with no ending punctuation.
- Assign a unique **access key** to each label. For assignment guidelines, see **Keyboard**.
- Use **sentence-style capitalization**.
- Position the slider label either to the left of the slider, or above and aligned with the left edge of the slider (or its left range identifier, if present).

3.13.17.2 Range Labels

- Label the two ends of the slider range, unless a vertical orientation makes this unnecessary.
- Use only word, if possible, for each label.
- Don't use ending punctuation.
- Make sure these labels are descriptive and parallel. Examples: Maximum/Minimum, More/Less, Low/High, Soft/Loud.
- Use sentence-style capitalization.
- Don't assign access keys.

3.13.17.3 Value Labels

- If you need a value label, display it below the slider.
- Center the text relative to the control and include the units (such as pixels).

3.14 Spin Controls (Page 168)

3.14.1 Is This The Right Control?

To decide, consider these questions:

- **Is the control used for numeric input?** If not, use another control, such as a **drop-down list** or **slider**, to select from a fixed set of values. Use **scroll bars** for scrolling.
- **Do users think of the value as a relative quantity, not a numeric value?** If so, use a slider instead. Use spin boxes only for exact, known numeric values. For example, users think about setting their audio volume to low or medium—not about setting the value to 2 or 5.
- **Is the control paired with a text box?** If not, don't use. Spin controls shouldn't be used alone or with other types of controls besides a text box.
- **Are contiguous value ranges valid?** If not, use a drop-down list of valid values instead.
- **Is using the spin control practical?** Using a spin control is practical for:
  - Entering a small number, typically under 100.
  - Making small changes to an existing or default value.
  - While spin controls can be used for any numeric input, they are inefficient in situations other than these.
- **Is the spin control helpful?** Is the control used in a context where users are likely to be using their mouse? If not, consider a spin control optional.

- **Are the sibling controls drop-down lists?** If there are other drop-down lists, consider using a drop-down list for consistency.
- **Are touch or pen users a primary target?** If so, consider using a drop-down list instead. The arrow buttons in a spin control are too small to be used efficiently with touch or a pen.

If a slider or a spin box is possible, use a spin box if:

- Screen space is tight.
- A user is likely to prefer using the keyboard.

Use a slider if:

- Users will benefit from instant feedback.

3.14.2 Guidelines

3.14.2.1 General

Use spin controls whenever they are practical and helpful. Exception: To be consistent with other text boxes on the same user interface (UI), use spin controls even if they aren't always practical.

- Always make a spin control the "buddy" of the text box. Doing so places the spin control inside the text box. Correct: Incorrect: *In the*
- **Disable a spin control when its associated text box is disabled.** The spin control is a supplemental input method—never the only input method.

3.14.2.2 Values

- **Define the top button to increase the value by one unit and the bottom button to decrease by one unit.** Typically, the unit is one, but it should be the smallest common change in value. Ideally, the spin control should cover all valid values, and it should be more convenient than typing in the text.
  - **Use the spin control to limit input to valid values.** Using a spin control should never result in an incorrect value.
    - **At the end of a range of valid values, restart the range.** The spin control metaphor is that the user is spinning a wheel of values, hence this wheel-like behavior. **Exception:** Don't restart the range if the resulting value is certain to be incorrect.
  - **Use text instead of special numeric values.** Allow users to spin to these special values instead of having to know them and type them in.
  - **If the value has delimiters, the associated text box should have multiple input focus points.** Doing so allows the numeric segments to be manipulated individually.

– **If the value has units, use the spin control to change those units as well.**

3.14.2.3 Labels

Apply the text box labeling guidelines to label the associated text box. Spin controls are never labeled directly.

3.14.2.4 Status Bars

- Is the status relevant when users are actively using other programs? If so, use a notification area icon.
- Does the status item need to display notifications? If so, you must use a notification area icon.
- Is the window a primary window? If not, don't use a status bar. Dialog boxes, wizards, control panels, and property sheets shouldn't have status bars.
- Is the information primarily status? If not, don't use a status bar. Status bars must not be used as a secondary menu bar or toolbar.
- Does the information explain how to use the selected control? If so, display the information next to the associated control using a supplemental explanation or instruction label instead.
- Is the status useful and relevant? That is, are users likely to change their behavior as a result of this information? If not, either don't display the status, or put it in a log file.
- Is the status critical? Is immediate action required? If so, display the information in a form that demands attention and cannot be easily ignored, such as a dialog box or within the primary window itself
- Is the program intended primarily for novice users? Inexperienced users are generally unaware of status bars, so reconsider the use of status bars in this case.
- Making sure that the status information is useful and relevant. If not, don't provide a status bar at all.
- Not using status bars for crucial information. Users should never have to know what is in the status bar. If users must see it, don't put it in a status bar.

If you do only one thing:. Make sure that the status bar information is useful and relevant but not crucial.

3.14.3 Guidelines

3.14.3.1 General

- Consider providing a View Status Bar command if only some users will need the status bar information. Hide the status bar by default if most users won't need it.

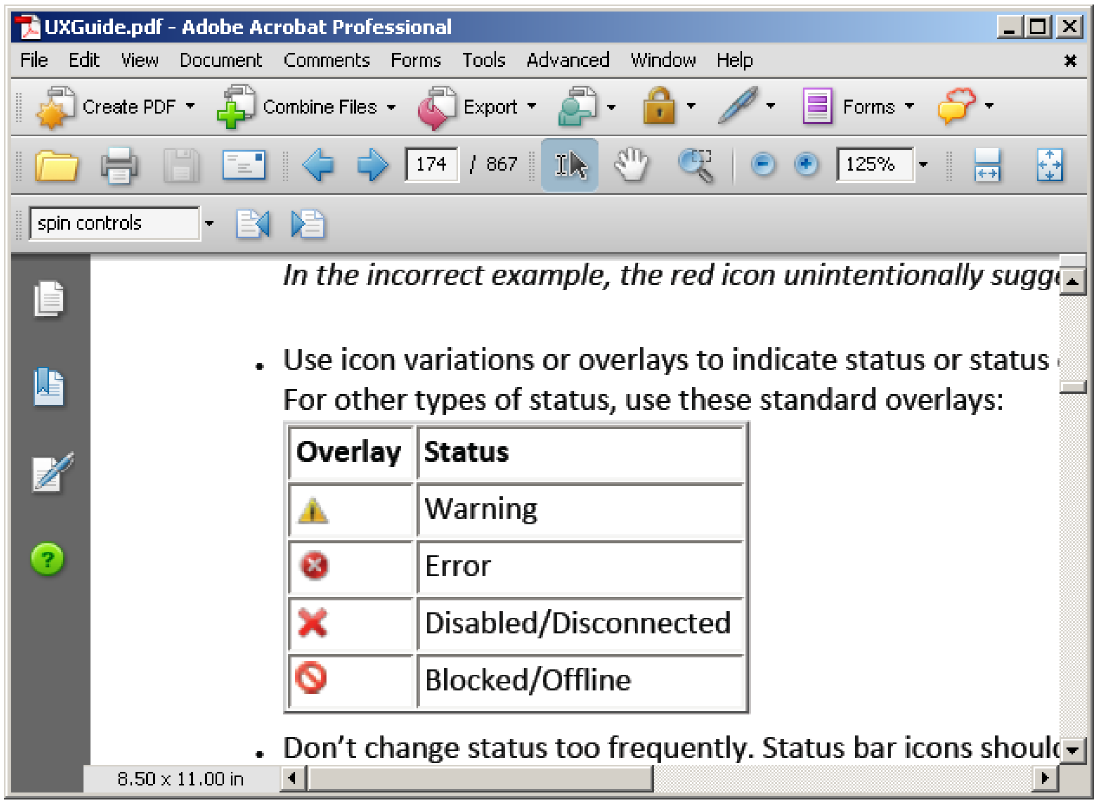
- Don't use the status bar to explain menu bar items. This help pattern isn't discoverable.

3.14.3.2 Presentation

- Disable modal status that doesn't apply. Modal status includes keyboard and document states.
- Remove non-modal status that doesn't apply.
- Present status information in the following order: current window status; progress; and contextual information.

3.15 Icons (Page 174)

- Choose easily recognizable status icon designs. Prefer icons with unique outlines over square or rectangular shaped icons.
- Use swaths of pure red, yellow, and green only to communicate status information. Otherwise, such icons are confusing.
  - Use icon variations or overlays to indicate status or status changes. Use icon variations to show changes in quantities or strengths. For other types of status, use these standard overlays:


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools Advanced Window Help
Create PDF
Combine Files
Export
Forms
174 / 867
125%
spin controls
8.50 x 11.00 in

In the incorrect example, the red icon unintentionally sugg

- Use icon variations or overlays to indicate status or status
For other types of status, use these standard overlays:

Overlay	Status
⚠	Warning
⊗	Error
✗	Disabled/Disconnected
🚫	Blocked/Offline

- Don't change status too frequently. Status bar icons shoul

Figure 3-11: Displaying icons

  - Don’t change status too frequently. Status bar icons shouldn’t appear noisy, unstable, or demand attention. The eye is sensitive to changes in the peripheral field of vision, so status changes need to be subtle.
  - For icons that provide important status information, prefer in-place labels.
  - Unlabeled status bar icons should have tooltips.

3.15.1 Interaction

Make a status bar area interactive to allow users direct access to related commands and options. Use a control that looks and behaves like a menu button or a split button. These status bar areas must have a drop-down arrow to indicate that they are clickable.

Display the menu on left-click on mouse down, not mouse up.

Don’t support right-clicking or double-clicking. Users don’t expect such interactions in a status bar, so they aren’t likely to attempt them.

Display tooltips on hover.

3.15.2 Text

- Generally, use concise labels. Cut any text that can be eliminated.
- Prefer sentence fragments, without ending punctuation. Use full sentences (with ending punctuation) only when sentence fragments aren’t significantly shorter.
- For optional progress labels, indicate what the operation is doing with a label that starts with a verb (gerund form) and ends with an ellipsis. For example: “Copying...”. This label may change dynamically if the operation has multiple steps or is processing multiple objects.
- Don’t use color, bold, or italic to emphasize status bar text.
- For tooltip phrasing guidelines, see **Tooltips and Infotips**.

3.16 Tabs (Page 176)

- Can the controls comfortably fit on a single, reasonably sized page? If so, use a single page.
- Is there only one tab? If so, use a single page.
- Are the tabs related to each other in some obvious way? If not, consider splitting the information into separate windows of related information.
- If used for settings, are settings on different pages completely independent? Will changing a setting on one page affect settings on other pages? If they’re not independent, use task pages or a wizard instead.

- Are the tabs mostly peers of each other, or is there a hierarchical relationship? If hierarchical, consider using progressive disclosure or child dialog boxes to show related information.
- Are the tabs used to display steps within a task? You can use "tabs" to display steps within a task only if they are presented to look like steps, and there is an obvious, alternative way to get to the text step, such as a Next button. Otherwise, if the steps are required, use pages in a page flow or a wizard. If the steps are optional, display the optional steps using modal dialog boxes instead.
- Are the tabs different views of the same data? If so, consider using a split button or drop-down list to change views. While tabs can be used effectively for changing views, the alternatives are more lightweight.

3.17 Guidelines (Page 178)

3.17.1.1 General

Use horizontal tabs if:

- The window has seven or fewer tabs.
- **All the tabs fit on one row, even when the user interface (UI) is localized.**

Use vertical tabs if:

- The property window has eight or more tabs.
- **Using horizontal tabs would require more than one row.**
  - **Don't nest tabs or combine horizontal tabs with vertical tabs.** Instead, reduce the number of tabs, use only vertical tabs, or use another control such as a drop-down list.
  - **Don't scroll horizontal tabs.** Horizontal scrolling isn't readily discoverable. You may scroll vertical tabs, however.
  - For tabs on a resizable window or pane, put a scrollbar, when needed, on the page, not the window or pane. The tabs should always be visible and not scroll out of view.
  - **Make sure the tabs look like tabs and not another type of control.**

3.17.1.2 Interaction

- When controls apply only to a page, place them within the border of the tabbed page.
- When controls apply to the entire window, place them outside the tabbed page.

- Don't assign effects to changing tabs. Tabs must be accessible in any order. Changing the current tab should never have side effects, apply settings, or result in an error message.
- Don't assign a special meaning to the last tab selected. Tab selection is for navigation—the user's last tab selection isn't a setting.
- Don't make the settings on a page dependent on settings on other pages. Put any dependent settings on the same page instead.
- If users are likely to start with the last tab displayed, make the tab persist and select it by default. Make the settings persist on a per-window, per-user basis. Otherwise, select the first page by default.

3.17.1.3 Icons

- Don't put icons on tabs. Icons usually add unnecessary visual clutter, consume screen space, and often don't improve user comprehension. Only add icons that aid in comprehension, such as standard symbols.
  - **Exception:** You can use clearly recognizable icons if there might be insufficient space to display meaningful labels:
  - **Don't use product logos for tab graphics.** Tabs aren't for **branding**.

3.17.2 Dynamic Window Surface Pattern

- **Don't use scroll bars on tab pages.** Tabs function similarly to scroll bars—to increase the effective area of a window. One mechanism should be sufficient.
- **Use concise tab labels.** Use one or two words that clearly describe the content of the page. Longer labels consume screen space, especially when the labels are localized.
- **Use specific, meaningful tab labels.** Avoid generic tab labels that could apply to any tab, such as General, Advanced, or Settings.
- **If a tab doesn't apply to the current context and users don't expect it to, remove it.** Doing so simplifies the UI and users won't miss it.
  - If a tab doesn't apply to the current context and users might expect it to: Display the tab.
  - Disable the controls on the page.
  - Include text explaining why the controls are disabled.
  - Don't disable the tab, because doing so isn't self-explanatory and prohibits exploration. Users looking for a specific value would be forced to look on all other tabs.

3.17.2.1 Multiple Views And Documents Patterns

- Use the view or document names on tab labels.

- Avoid excessively long tab names. If necessary, either have a maximum name size or truncate the displayed tab label using ellipses. Longer labels consume screen space, especially when the labels are localized.
- If a tab doesn't apply to the current context, remove the tab.

3.17.2.2 Exclusive Options Pattern

Don't use this pattern! Use radio buttons or a drop-down list instead.

3.17.3 Labels

- Label tabs based on their pattern. Use nouns rather than verbs, without ending punctuation. See the preceding pattern guidelines for more information.
- Use **sentence-style capitalization**.
- Don't assign an **access key**. Tabs are accessible through their shortcut keys (Ctrl+Tab, Ctrl+Shift+Tab, Ctrl+PgUp, Ctrl+PgDn). There is a shortage of good access key choices, so not assigning access keys to tabs makes it easier to assign them to other controls.

3.18 Text Boxes (Page 185)

- Is it practical to enumerate all the valid values efficiently? If so, consider a single-selection list, list view, drop-down list, editable drop-down list, or slider instead.
- Is the valid data completely unconstrained? Or is the valid data constrained only by format (constrained length or character types)? If so, use a text box.
- Does the value represent a data type that has a specialized common control? Examples include date, time, or IPv4 or IPv6 address. If so, use the appropriate control, such as a date control rather than a text box.
- If the data is numeric: Do users perceive the setting as a relative quantity? If so, use a slider.
- Would the user benefit from instant feedback on the effect of setting changes? If so, use a slider, possibly along with a text box. For example, users can easily choose a color using a slider because they can immediately see the effect of changes to hue, saturation, or luminosity values.

3.18.1 Guidelines

3.18.1.1 General

- When disabling a text box, also disable any associated labels, instruction labels, spin controls, and command buttons.

- Use auto-complete to help users enter data that is likely to be used repeatedly. Examples include user names, addresses, and file names. However, don't use auto-complete for text boxes that may contain sensitive information, such as passwords, PINs, credit card numbers, or medical information.
- Don't make users scroll unnecessarily. If you expect data to be larger than the text box and you can readily make the text box larger without harming the layout, size the box to eliminate the need for scrolling.
  - Scroll bars: **Don't put horizontal scroll bars on multiline text boxes.** Use vertical scrolling and line wrapping instead.
  - **Don't put any scroll bars on single-line text boxes.**
    - **For numeric input, you may use a spin control.** For textual input, use a drop-down list or editable drop-down list instead.
    - **Don't use the auto-exit feature except for formatted data input.** The automatic shift of focus can surprise users.

3.18.2 Editable Text Boxes

- **Limit the length of the input text when you can.** For example, if the valid input is a number between 0 and 999, use a numeric text box that is limited to three characters. All parts of text boxes that use formatted data input must have a short, fixed length.
- **Be flexible with data formats.** If users are likely to enter text using a wide variety of formats, try to handle all the most common ones. For example, many names, numbers, and identifiers can be entered with optional spaces and punctuation, and the capitalization often doesn't matter.
- If you can't handle the likely formats, require a specific format by using formatted data input or indicate the valid formats in the label.
- Don't use the formatted data input pattern if users are more likely to paste in long, complex data. Rather, reserve the formatted data input pattern for situations where users are more likely to type the data.
- If users are more likely going to reenter the entire value, select all the text on input focus. If users are more likely to edit, place the caret at the end of the text.
- Always use a multiline text box if new-line characters are valid input.
- When the text box is for a file or path, always provide a Browse button.

3.18.3 Numeric Text Boxes

- **Choose the most convenient unit and label the units.** For example, consider using milliliters instead of liters (or vice versa), percentages instead of direct values (or vice versa), and so on.

- **Use a spin control whenever it is helpful.** However, sometimes spin controls aren't practical, such as when users need to enter many large numbers. Use spin controls when:
  - The input is likely to be a small number, typically under 100.
  - Users are likely to make a small change to an existing number.
  - Users are more likely to be using the mouse than the keyboard.
- **Right-align numeric text whenever:**
  - There is more than one numeric text box.
  - The text boxes are vertically aligned.
  - Users are likely to add or compare the values.
    - Always right-align monetary values.
    - Don't assign special meanings to specific numeric values, even if those special meanings are used internally by your application. Instead, use check boxes or radio buttons for an explicit user selection.

3.18.4 Password and PIN Input

- **Always use the password common control instead of creating your own.** Passwords and PINs require special treatment to be handled securely.

3.18.4.1 Data Output

- Don't use a border for single-line, read-only text boxes. The border is a visual clue that the text is editable.
- Don't disable single-line, read-only text boxes. This prevents users from selecting and copying the text to the clipboard. It also prevents users from scrolling the data if it exceeds the size of its boundaries.
- Don't set a tab stop on single-line, read-only text box unless the user is likely to need to scroll or copy the text.

3.18.5 Input Validation And Error Handling

- If the user enters a character that isn't valid, ignore the character and display an **input problem balloon** that explains the valid characters
  - If the input data has a value or format that isn't valid, display an input problem balloon when the text box loses input focus.
  - If the input data is inconsistent with other controls on the window, give an error message when the entire input is complete, such as when users click OK for a modal dialog box.

Don't clear invalid input data unless users aren't able to correct errors easily. Doing so allows users to correct mistakes without starting over. For example, you should clear incorrect passwords and PINs because users can't correct them easily.

3.19 Prompts (Page 190)

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 in a compact way. It must not be crucial information that the user needs to see while using the text box.
- Don't use prompts just to direct users to type something or to click buttons. For example, don't write prompt text that says *Enter a filename and then click Send*.
- Draw the prompt text in italic gray and the actual input text in normal black. The prompt text must not be confused with real text.
- Keep the prompt text concise. You can use fragments instead of full sentences.
- Use **sentence-style capitalization**.
- Don't use ending punctuation or ellipsis.
- 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.
  - The prompt text is restored if the text box is still empty when it loses input focus.

3.19.1 Recommended Sizing And Spacing

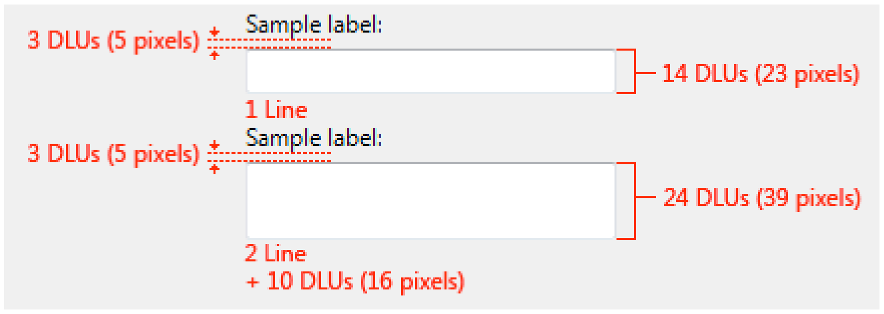


Figure 3-12: Recommended sizing and spacing of prompts

- **Choose a width appropriate for the longest valid data.** In most situations, users shouldn't have to scroll the longest likely string they'll enter or view.
- **Include an additional 30%** (up to 200% for shorter text) for any text (but not numbers) that will be localized.

- **If the expected input has no particular size, choose a width that is consistent with the other text boxes or controls on the window.**
- **Size multiline text boxes to display an integral number of lines of text.**

3.19.2 Labels

3.19.2.1 Text Box Labels

- All text boxes need labels. Write the label as a word or phrase, not as a sentence, ending with a colon, and using **static text**.

  Exceptions:

  - Text boxes with prompts located where space is at a premium.
  - For labeling, a group of text boxes used for formatted data input should be treated as a single text box.
  - If a text box is subordinate to a radio button or check box, and is introduced by its label ending with a colon, don't put an additional label on the text box.
  - Omit control labels that restate the main instruction. In this case, the main instruction takes the colon (unless it's a question) and access key.
    - Assign a unique **access key**. For access key assignment guidelines, see **Keyboard**.
    - Use **sentence-style capitalization**.
    - Position the label either to the left of or above the text box, and align the label with the left edge of the text box. If the label is on the left, vertically align the label text with the text box text.
    - You may specify units (for example, seconds or connections) in parentheses after the label.
    - If a text box accepts an arbitrarily small maximum number of characters, you can state the maximum input in the label. The text box width should also suggest the maximum size.
    - Don't make the content of the text box (or its units label) part of a sentence, because this is not localizable.
    - If the text box can be used to enter several items, make it clear how to separate the items in the label.

3.19.2.2 Instruction Labels

If you need to add instructional text about a text box, add it above the label.

- Use complete sentences with ending punctuation.
- Use **sentence-style capitalization**.

- Additional information that is helpful but not necessary should be kept short. Place this information either in parentheses between the label and colon, or without parentheses below the text box.

3.19.2.3 Prompt Labels

- Keep the prompt text concise. You can use fragments instead of full sentences.
- Use **sentence-style capitalization**.
- Don’t use ending punctuation or ellipsis.
- If the prompt directs users to enter information that will be acted upon by a button next to the text box, simply place the button next to the text box. Don’t use the prompt to direct users to click the button (for example, don’t write prompt text that says, *Drag a file and then click Send*).

3.20 Tooltips and Infotips (Page 194)

Developers: There is no infotip control; infotips are implemented with the tooltip control. The distinction is in usage, not implementation.

- **Is the information displayed based on pointer hover?** If not, use another control. Display tips only as the result of user interaction—never display them on their own. By contrast, **balloons** can display on their own (as they do with notifications), so they have a tail that identifies their source.
- **Does a control have a text label?** If not, use a tooltip to provide the label. Note that most controls should be labeled and therefore not have tooltips. Toolbar controls and command buttons with graphic labels should have tooltips.
  - **Does an object benefit from a supplemental description or further information?** If so, use an infotip. However, the text must be supplemental—that is, not essential to the primary tasks. If it is essential, put it directly in the UI so that users don’t have to discover or hunt for it.
  - **Is the supplemental information an error, warning, or status?** If so, use another UI element, such as a balloon, **error message**, or **status bar**. Notification area icon infotips are an exception because they can be used to show status information.
  - **Do users need to interact with the tip?** If so, use another control, such as a balloon. Users can’t interact with tips because moving the mouse makes them disappear.
  - **Do users need to print the supplemental information?** If so, use another control, such as a static comment field. However, you can also use infotips to provide more direct access to this information.
  - **Is the context such that users might find the tips annoying or distracting?** If so, consider using another solution—including doing nothing at all. If you do use tips in such contexts, allow users to turn them off.

3.20.1 Appropriate Timeouts

The appropriate automatic display and removal of tips is crucial to the goal of users maintaining control of their UI environment. Tips have three timeout values:

- **Initial.** The time the pointer must remain stationary for the tip to appear. The default time is 0.5 seconds.
- **Reshow.** The time the pointer must remain stationary as the pointer moves from one target to another. The default time is 0.1 seconds.
- **Removal.** The time after which the tip is automatically removed. The default time is 5 seconds.

If you do only one thing: Design discoverable tips that display concise, helpful, static, supplemental information in the appropriate place at the appropriate time.

3.21 Guidelines (Page 200)

3.21.1 Timeouts

Use the default initial and reshow timeouts. Exception: Thumbnails that aren't redundant and displayed on the side of their associated object can be shown immediately (without any delay). However, use the default initial timeout for redundant thumbnails (such as a large thumbnail tip for a small graphic object) or thumbnails that cover their associated object.

- For tooltips, use the default five-second tip removal timeout.
- For infotips, turn off the tip removal timeout. Developers: Since you can't technically turn off the removal timeout, set it to its largest value.
- For accessibility, if you need to set the timeout values to something other than the maximum value, make them multiples of the SPI_GETMOUSEHOVERTIME and SPI_GETMESSAGEDURATION system parameters instead of using fixed times. Doing so adjusts the timeouts to the speed of the user.

3.21.2 Placement

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.

- **For potentially distracting (often large) tips, make sure that the information is helpful for most users.** If that's not the case, either make the distracting tips optional or even eliminate them. Otherwise, most users will have to move the pointer away from the target object to get rid of the tip.

3.21.3 Tooltips

- **Use tooltips to provide labels for unlabeled controls.** Controls that commonly have tooltips are **toolbar buttons**, graphic buttons, and **progressive disclosure controls**. Controls with prompts are considered labeled, such as **text boxes** and **combo boxes**. All other controls should have explicit labels.
- Use sentence fragments without ending punctuation.
  - Use **sentence-style capitalization**. **Exception:** This guideline is new for Windows Vista®. For legacy applications, you may use **title-style capitalization** if necessary to avoid mixing capitalization styles.
  - Add an **ellipsis** if the label is for a command that needs additional information.
  - As with normal labels, **keep tooltips brief**—typically five words or less—but prefer specific labels over vague ones.
    - 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.
    - You don't have to give labeled controls tooltips simply for the sake of consistency
    - Whenever appropriate, make tooltips more helpful by providing keyboard shortcuts and default values. Put this additional information in parentheses. Doing so makes tooltips helpful for labeled controls even when they otherwise just repeat the label. Don't consider this additional text when evaluating the conciseness of a tooltip.

3.21.4 Infotips

- **For infotips in nonstandard places, favor consistency over helpfulness to improve discoverability.** Provide tips for all objects for which users are likely to want supplemental information, even if a few infotips might be obvious. Doing so avoids having users wait for an infotip that will never come. **Exception:** If only a few objects have helpful infotips, don't use infotips at all. Rather, use self-explanatory control labels or in-place supplemental text instead.
- Use full sentences with ending punctuation. **Exception:** Notification area **icon infotips** don't use ending punctuation.
- Use **sentence-style capitalization**.
- Use present tense, not future.

- Use parallel grammatical constructions. Parallelism requires that words and phrases that have the same function have the same form. **Exception:** For the **full name infotip** pattern, the infotip text exactly matches the phrasing, capitalization, and punctuation of the underlying control.
- **Avoid large infotips.** Large infotips are difficult to read, and difficult to position without interfering with the underlying object.
- **Format infotips to make their content easier to read and scan.** Large blocks of unformatted text can be difficult to read.

3.21.4.1 Start Menu Infotips

- Use Start menu infotips to describe the item concisely and list the primary tasks that users can perform with the item.
- Be helpful. Focus on what users can do. Don't just repeat the item name or even use it in the description at all.
- Be specific. Avoid generic verbs and catch-all phrases like *and other tasks*. If the information is important, list it specifically; otherwise, assume that users understand that not everything is listed in the infotips.
- Be concise. Use 25 words or less. Longer infotips discourage reading.
- Start with a present-tense, imperative verb such as *create*, *edit*, *show*, and *send*. Prefer specific verbs over generic verbs such as *manage* and *open*, which really apply to most Start menu items. Get right to the point.
  - Don't use language that sounds like marketing.
  - Because these infotips are indexed for the Start menu search box, describe your program's important tasks using terms for which users are most likely to search. Consider using keywords and common synonymsUse sentence-style capitalization.
- **Developers:** The Start menu infotip text comes from the item's Comment field.

3.21.4.2 Quick Launch Tooltips

- Use a tooltip with the format: Launch (full program name)
- Don't use ending punctuation.
- Don't use additional text to describe the program or what it does. Because users choose the programs displayed in the Quick Launch bar, they already know their purpose.

3.21.4.3 Control Panel Infotips

- Use Control Panel infotips to concisely describe the Control Panel tasks and the hardware and software configured.

- Control Panel names and icons must have infotips. Individual tasks don't have tooltips.
- Be helpful. Focus on what users can do. Don't just repeat the Control Panel item name or even use it in the description at all.
- Be specific. Avoid generic verbs and catch-all phrases like *and other hardware*. If the information is important, list it specifically; otherwise, assume that users understand that not everything is listed in the infotips.
  - **Be concise.** Use 25 words or less. Longer infotips discourage reading.
  - **Start with a present-tense, imperative verb.**
  - **Get right to the point.** Don't use language that applies to any Control Panel, such as "Use to view and configure settings for the appearance and functionality of your..." or "Provides options for you to..."
  - Don't use language that sounds like marketing.
  - Because these infotips are indexed for the Control Panel search box, **describe items using terms for which users are most likely to search.** Consider using common synonyms for popular tasks and objects.
  - If a Control Panel item is likely to be confused with others, explain how it is different in the infotip.

3.21.5 Icons

Unlike previous versions of Windows, Windows Vista allows tips to have icons.

- For tooltips, don't use icons.
- For infotips, use icons only if they aid in recognition or comprehension, or provide context. Most infotips shouldn't have icons.
- The icon must use the Aero-style and have an unobtrusive appearance.

3.22 Tree Views (Page 208)

- **Is the data hierarchical?** If not, use another control.
- **Does the hierarchy have at least three levels (not including the root)?** If not, consider alternatives such as list view groups, tabs, drop-down lists, or expandable headings.
- **Do the items have auxiliary data?** If so, consider using a list view in the Details view mode to take full advantage of the auxiliary data.
- **Does the lower-level data relate to independent subtasks?** If so, consider displaying the information in an associated control or in a separate window (displayed using **command buttons** or **links**).

- **Are the target users advanced?** Advanced users are more proficient at using trees. If your application is aimed at novice users, avoid using tree views.
- **Do the items have a single, natural, hierarchical categorization that's familiar to most users?** If so, the data is ideal for a tree view. If there is a need for multiple views or sorting, use a list view instead.
- **Do users need to see the lower-level data in some but not all scenarios, or some but not all of the time?** If so, the data is ideal for a tree view.

3.22.1 Guidelines

3.22.1.1 Presentation

- **Within a container, sort the items in a logical order.** Sort names in alphabetical order, numbers in numeric order, and dates in chronological order.
- **Use the Always Show Selection attribute** so that users can readily determine the selected item, even when the control doesn't have input focus.
- **If the tree is acting as a table of contents, use the Single Expand attribute to simplify the management of the tree.** This way, only the relevant portion of the tree is expanded.
- **Avoid presenting empty trees.** If a user creates a tree, initialize the tree with instructions or example items that users might need.
  - Don't make the container nodes collapsible if users have no reason to collapse them. Doing so adds unnecessary complexity.
  - If load performance is a problem, display only the first and second level containers of the tree by default. You can then load additional data on demand when a user expands branches in the tree.
  - If users expand or collapse a container, make that state persist so it takes effect the next time the tree view is displayed, unless users are likely to prefer starting in the default state. Persistence should be on a per-tree view, per-user basis.
  - If high-level containers have similar contents, consider using visual clues to differentiate them.

3.22.1.2 Interaction

- **Consider providing double-click behavior.** Double-clicking should have the same effect as selecting an item and performing its default command.
- **Make double-click behavior redundant.** There should always be a command button or context menu command that has the same effect.
- If an item requires further explanation, **provide the explanation in an infotip**.

- **Provide context menus of relevant commands.** Such commands include Cut, Copy, Paste, Remove or Delete, Rename, and Properties.
- **When disabling a tree view, also disable any associated labels and command buttons.**

3.22.2 Tree Organization

- Use a natural hierarchical structure that's familiar to most users.
- If you can't use such a structure, try to balance discoverability with a predictable user model that minimizes confusion.
  - **To safely improve discoverability, place an item in multiple containers when:** The item isn't related to any other similar items (so users don't become confused by incorrect associations).
  - There are only a few of such redundantly located items (so the tree doesn't become bloated).
  - **Use the simplest hierarchical structure that works well.** To do so: Place the most commonly accessed objects in the first two levels of the tree (not counting the root node), and place less commonly accessed objects farther down the hierarchy.
  - Eliminate unnecessary or combine redundant intermediate-level containers.
  - **Prefer breadth over depth.** Ideally, a tree should have no more than four levels and the most commonly accessed objects should appear in the first two levels.
  - **Determine if you really need a root node.** Provide a root node if users need the ability to perform commands on the entire tree (possibly using a context menu on the root node). Otherwise, the tree is simpler and easier to use without it.
- If the tree has alternative access methods such as a word search or an index, optimize the tree for browsing by focusing on the most useful content. With alternative access methods, the tree's content doesn't have to be comprehensive. Simplifying the tree makes it easier for users to find the most useful content.

3.22.2.1 Check Box Tree Views

- **Display the number of selected items below the list**, especially if users are likely to select several items. This feedback helps users confirm that their selection is correct.
- If there are potentially many items and selecting or clearing all of them is likely, add Select all and Clear all command buttons.
- **Use mixed-state check boxes to indicate partial selection of the items in a container.**

3.22.2.2 Recommended Sizing And Spacing

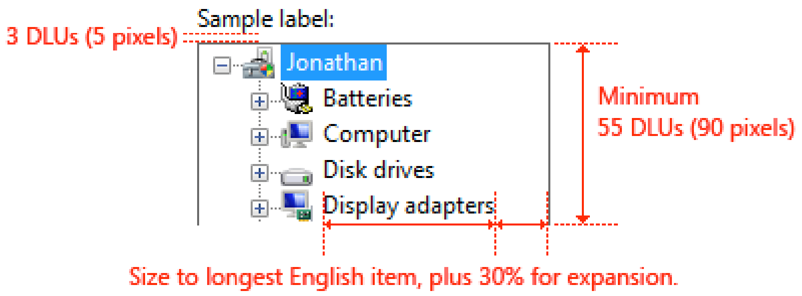


Figure 3-13: Recommended sizing and spacing of check box trees

- **Choose a tree view width that avoids the need for horizontal scrolling** for most items when the tree is fully expanded.
- **Include an additional 30% to accommodate localization.**
- **Choose a tree view height that eliminates unnecessary vertical scrolling.** Consider making a tree view slightly longer (or even more so if there is available space) if doing so reduces the need for a vertical scroll bar.
- **If users benefit from making the tree view larger, make the tree view and its parent window resizable.** Doing so allows users to adjust the tree view size as needed.

3.22.3 Labels

3.22.3.1 Control Labels

- All tree views need labels. Write the label as a word or phrase, not as a sentence, ending with a colon, and using **static text**.
- Assign a unique **access key**. For assignment guidelines, see **Keyboard**.
- Use **sentence-style capitalization**.
- Position the label above the control, and align the label with the left edge of the control.
- For multiple-selection

3.22.4 Data Text

Use sentence-style capitalization.

3.22.5 Instructional Text

- If you need to add instructional text about a tree view, add it above the label. Use complete sentences with ending punctuation.

- Use sentence-style capitalization.
- Supplemental explanations that are helpful but not necessary should be kept short. Place this information either in parentheses between the label and colon, after the main instruction if used instead of a label, or below the control.

3.23 Commands (Page 221)

3.23.1 Menus

3.23.1.1 Guidelines

3.23.1.1.1 General

All menu patterns except menu bars need a drop-down arrow to indicate the presence of a pull-down menu. The presence of menus goes without saying in a menu bar, but not in the other patterns.

Don't change menu item names dynamically. Doing so is confusing and unexpected. For example, don't change a Portrait mode option to Landscape mode upon selection. For modes, use bullets and checkmarks instead. Exception: You can change menu item names that are based on object names dynamically. For example, lists of recently used files or window names can be dynamic.

3.23.1.2 Menu Bars

- **Consider eliminating menu bars with three or fewer menu categories.** If there are only a few commands, prefer lighter alternatives such as toolbar menus, or more direct alternatives such as command buttons and links.
- **Don't have more than 10 menu categories.** Too many menu categories is overwhelming and makes the menu bar difficult to use.
- **Consider hiding the menu bar** if the toolbar or direct commands provide almost all of the commands needed by most users. Allow users to show or hide with a Menu bar check mark option in a toolbar menu.

3.23.1.3 Hiding Menu Bars

Generally, toolbars work great together with menu bars because having both allows each to focus on their strengths without compromise.

- Hide the menu bar by default if your toolbar design makes having a menu bar redundant.
- Hide the menu bar instead of removing it completely, because menu bars are more accessible for keyboard users.

- To restore the menu bar, provide a Menu bar checkmark option in the View (for primary toolbars) or Tools (for secondary toolbars) menu category. For more information, see **Standard menu and split buttons**.

3.23.1.4 Menu Categories

- **Choose single word names for menu categories.** Using multiple words makes the separation between categories confusing.
- **For programs that create or view documents, use the standard menu categories** such as File, Edit, View, Tools, and Help. Doing so makes common menu items predictable and easier to find.
- **For other types of programs, consider organizing your commands and options into more useful, natural categories** based on your program's purpose and the way users think about their tasks and goals. Don't feel obligated to use the standard menu organization if it isn't suitable for your program.
- **If you choose to use non-standard menu categories, you must choose good category names.** For more information, see the **Labels** section.
- **Prefer task-oriented menu categories over generic categories.** Task-oriented categories make menu items easier to find.
  - **Avoid menu categories with only one or two menu items.** If sensible, consolidate with other menu categories, perhaps using a submenu.
    - Consider putting the same menu item in multiple categories only if: The menu item logically belongs in multiple menu categories.
    - You have data showing that users have trouble finding the item in a single menu category.
    - You have only one or two hard-to-find menu items in multiple categories.
    - Don't put different menu items that use the same name in multiple categories. For example, don't have different Options menu items in multiple categories. Exception: The tab menu pattern may have different Options and Help menu items in each tab menu.

3.23.1.5 Menu Item Organization And Order

- Organize the menu items into groups of seven or fewer strongly related items. For this, submenus count as a single menu item in the parent menu.
- Don't put more than 25 items within a single level of a menu (not counting submenus).
- Put separators between the groups within a menu. A separator is a single line that spans the width of the menu.
- Within a menu, put the groups in their logical order. If there is no logical order, place the most commonly used groups first.

- Within a group, put the items in their logical order. If there is no logical order, place the most commonly used items first. Put numeric items (such as zoom percentages) in numeric order.

3.23.1.6 Submenus

- **Avoid using submenus unnecessarily.** Submenus require more physical effort to use and generally make the menu items more difficult to locate.
- **Don't put frequently used menu items in a submenu.** Doing so would make using these commands inefficient. However, you can put frequently used commands in a submenu if they are normally accessed more directly, such as with a toolbar.
- Consider using a submenu if:
  - Doing so simplifies the parent menu because it has many items (20 or more), or the submenu is part of a group of more than seven items.
  - The items in the submenu are used less frequently than those in the parent menu.
  - The submenu would have three or more items.
  - There are three or more commands that begin with the same word. In this case, use that word as the submenu label.
    - **Use at most three levels of menus.** That is, you can have a primary menu and at most two levels of submenus. Two levels of submenus should be rare.

3.23.1.6.1 Presentation

- Disable menu items that don't apply to the current context, instead of removing them. Doing so makes menu bar contents stable and easier to find. Exceptions: For contextual menu categories, remove rather than disable context menu items that don't apply to the current context. A menu category is contextual when it is displayed only for specific modes, such as when a certain object type is selected. For details, see the remove vs. disable guidelines for context menus.
- If determining when a menu item should be disabled causes noticeable performance problems, leave the menu item active and if necessary have its selection result in an error message.

3.23.1.7 Tab Menus

Each tab menu may have context specific Options and Help menu items. This is in contrast to all other menu patterns. Each tab is used for a dedicated set of tasks, so any redundancy across tab menus isn't confusing.

3.23.1.8 Context Menus

- **Use context menus only for contextual commands and options.** The menu items should apply only to the selected (or clicked upon) object or window region, not the entire program.
- **Don't make commands only available through context menus.** Like shortcut keys, context menus are alternative means of performing commands and choosing options. For example, a Properties command is also available through the menu bar or the Alt+Enter access key.
- **Provide context menus for all objects and window regions** that benefit from a small set of contextual commands and options. Many users right-click regularly and expect to find context menus anywhere.
- **Consider using a menu drop-down arrow button for context menus targeted at all users.** Normally context menus are suitable for commands and options targeted at advanced users. However, you can use a menu drop-down button in cases where context menus are the best menu choice and you need to target all users.

3.23.1.9 Menu Item Organization And Order

- Organize the menu items into groups of seven or fewer strongly related items.
- Avoid using submenus to keep context menus simple, direct, and efficient.
- Don't put more than 15 items within a context menu.
- Put separators between the groups within a menu. A separator is a single line that spans the width of the menu.
- Present menu items using the following order: Primary (most frequently used) commands Open Run Play Print <separator> Secondary commands supported by the object <separator> Transfer commands Cut Copy Paste <separator> Object settings <separator> Object commands Delete Rename <separator> Properties

3.23.1.9.1 Presentation

- **Display the default command using bold.** When practical, also make it the first menu item. The default command is invoked when users double-click or select an object and press Enter.
- **Remove rather than disable context menu items that don't apply to the current context.** Doing so makes context menus contextual and efficient. **Exception:** Disable menu items that don't apply if there is a reasonable expectation for them to be available: Always have the relevant standard context menu commands, such as Cut, Copy, Paste, Delete, and Rename.
- Always have the commands that complete related sets. For example, if there is a Back, there should also be a Forward. If there's a Cut, always have a Copy and Paste.

3.24 Bullets and Checkmarks (Page 232)

- Menu items that are options may use bullets and checkmarks. Commands may not.
- Use a bullet to choose one option from a small set of mutually exclusive choices. There should always be at least two bullets in a group. For more information, see Radio buttons.
- Use a checkmark to toggle an independent setting on or off. If the selected and cleared states aren't clear and unambiguous opposites, use a set of bullets instead. For more information, see Check boxes.
- For a mixed checkmark state, display a menu item without a checkmark. The mixed state is used for multiple selection to indicate that the option is set for some, but not all, objects, so each individual object has either the selected or cleared state. The mixed state is not used as a third state for an individual item.
- Put separators between the related sets of checkmarks or bullets. A separator is a single line that spans the width of the menu.

3.24.1 Icons

Consider providing menu item icons for:

- The most commonly used menu items.
- Menu items whose icon is standard and well known.
- Menu items whose icon well illustrates what the command does.

If you use icons, don't feel obligated to provide them for all menu items. Cryptic icons aren't helpful, create visual clutter, and prevent users from focusing on the important menu items.

Make sure menu icons conform to the Aero-style icon guidelines.

3.24.2 Access Keys

- Assign access keys to all menu items. No exceptions.
- Whenever possible, assign access keys for commonly used commands according to the Standard Access Key Assignments. While consistent access key assignments aren't always possible, they are certainly preferred—especially for frequently used commands.
- For dynamic menu items (such as recently used files), assign access keys numerically.
  - **Assign unique access keys within a menu level.** You can reuse access keys across different menu levels.

- **Make access keys easy to find:** For the most frequently used menu items, choose characters at the beginning of the first or second word of the label, preferably the first character.
- For less frequently used menu items, choose letters that are a distinctive consonant or a vowel in the label.
- **Prefer characters with wide widths,** such as w, m, and capital letters.
- **Prefer a distinctive consonant or a vowel,** such as “x” in “Exit.”
- **Avoid using characters that make the underline difficult to see,** such as (from most problematic to least problematic): Letters that are only one pixel wide, such as “I” and “l”.
- Letters with descenders, such as “g”, “j”, “p”, “q”, and “y”.
- Letters next to a letter with a descender.

3.24.3 Shortcut Keys

- **Assign shortcut keys to the most frequently used menu items.** Infrequently used menu items 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.
- **For well-known shortcut keys, use the standard assignments.** See **Windows Keyboard Shortcut Keys** for the well-known shortcut keys used by Windows programs.
- **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. See Windows Keyboard Shortcut Keys for the well-know shortcut keys used by Windows programs.
- **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.
- **Document all shortcut keys.** Doing so helps users learn the shortcut key assignments. **Exception:** Don’t display shortcut key assignments within context menus. Context menus don’t display the shortcut key assignments because they are optimized for efficiency.
- **For non-standard key assignments: Choose shortcut keys that don’t have standard assignments.** Never reassign standard shortcut keys.
- **Use nonstandard key assignments consistently throughout your program.** Don’t assign different meanings in different windows.
- **If possible, choose mnemonic key assignments,** especially for frequently used commands.

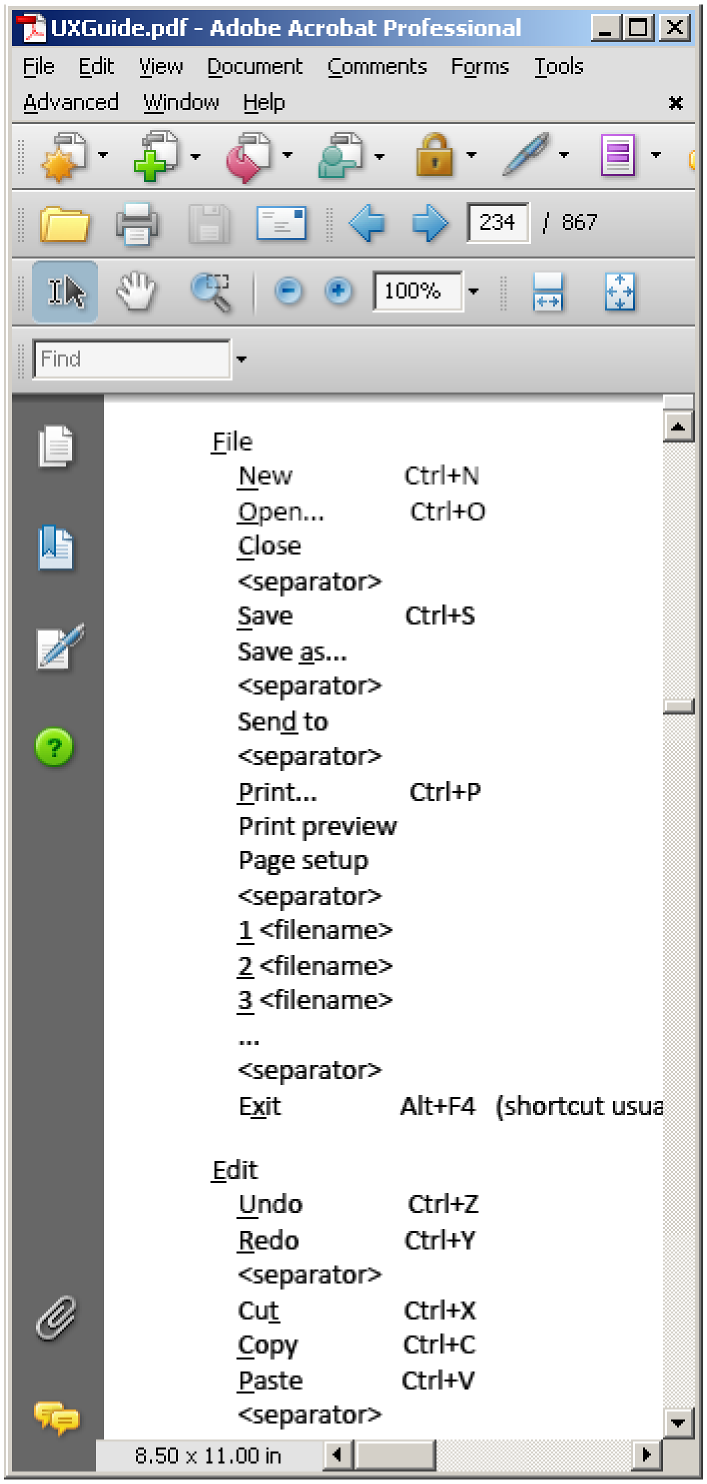
- **Use function keys for commands that have a small-scale effect,** such as commands that apply to the selected object. For example, F2 renames the selected item.
- **Use Ctrl key combinations for commands that have a large-scale effect,** such as commands that apply to an entire document. For example, Ctrl+S saves the current document.
- **Use Shift key combinations for commands that extend or complement the actions of the standard shortcut key.** For example, the Alt+Tab shortcut key cycles through open primary windows, whereas Alt+Shift+Tab cycles in the reverse order. Similarly, F1 displays Help, whereas Shift+F1 display context-sensitive Help.
- **Don’t use the following characters for shortcut keys:** @ £ $ {} [] \ ~ | ^ ' < >. These characters require different key combinations across languages or are locale specific.
  - **Don’t use Ctrl+Alt combinations,** because Windows interprets this combination in some language versions as an AltGR key, which generates alphanumeric characters.
  - **If your program assigns many shortcut keys, provide the ability to customize the assignments.** Doing so allows users to reassign conflicting shortcut keys and migrate from other products. Most programs don’t assign enough shortcut keys to need this feature.

3.24.4 Standard Menus

- **Use the standard menu organization for programs that create or view documents.** The standard menu organization makes common menu items predictable and easier to find.
- **For other types of programs, use the standard menu organization only when it makes sense to.** Consider organizing your commands and options into more useful, natural categories based on your program’s purpose and the way users think about their tasks and goals.

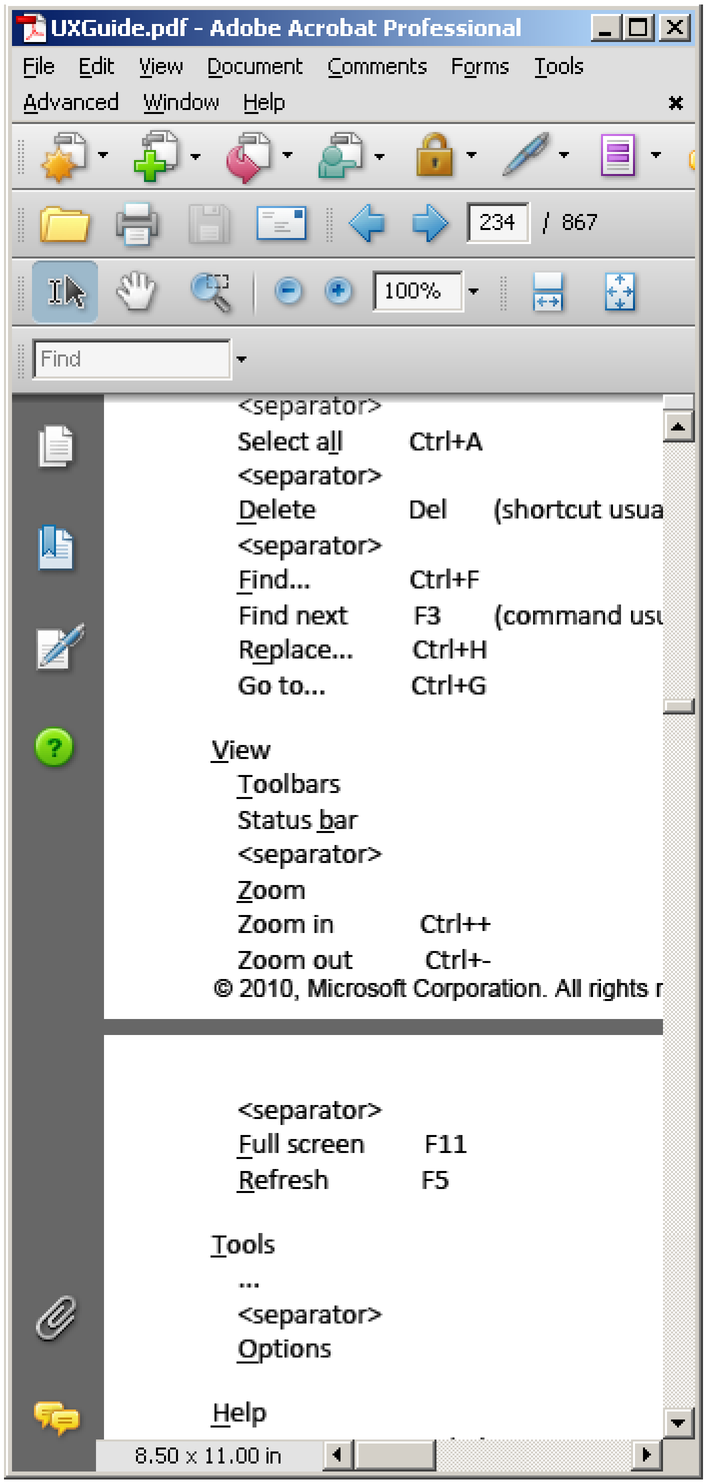
3.24.4.1 Standard Menu Bars

The standard menu bar structure is as follows. This list shows the menu category and item labels, their order with separators, their access and shortcut keys, and their ellipses.


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
234 / 867
100%
Find
8.50 x 11.00 in

File		
New	Ctrl+N	
Open...	Ctrl+O	
Close		
<separator>		
Save	Ctrl+S	
Save as...		
<separator>		
Send to		
<separator>		
Print...	Ctrl+P	
Print preview		
Page setup		
<separator>		
1 <filename>		
2 <filename>		
3 <filename>		
...		
<separator>		
Exit	Alt+F4	(shortcut usua
Edit		
Undo	Ctrl+Z	
Redo	Ctrl+Y	
<separator>		
Cut	Ctrl+X	
Copy	Ctrl+C	
Paste	Ctrl+V	
<separator>		

Figure 3-14: Menu category and item labels

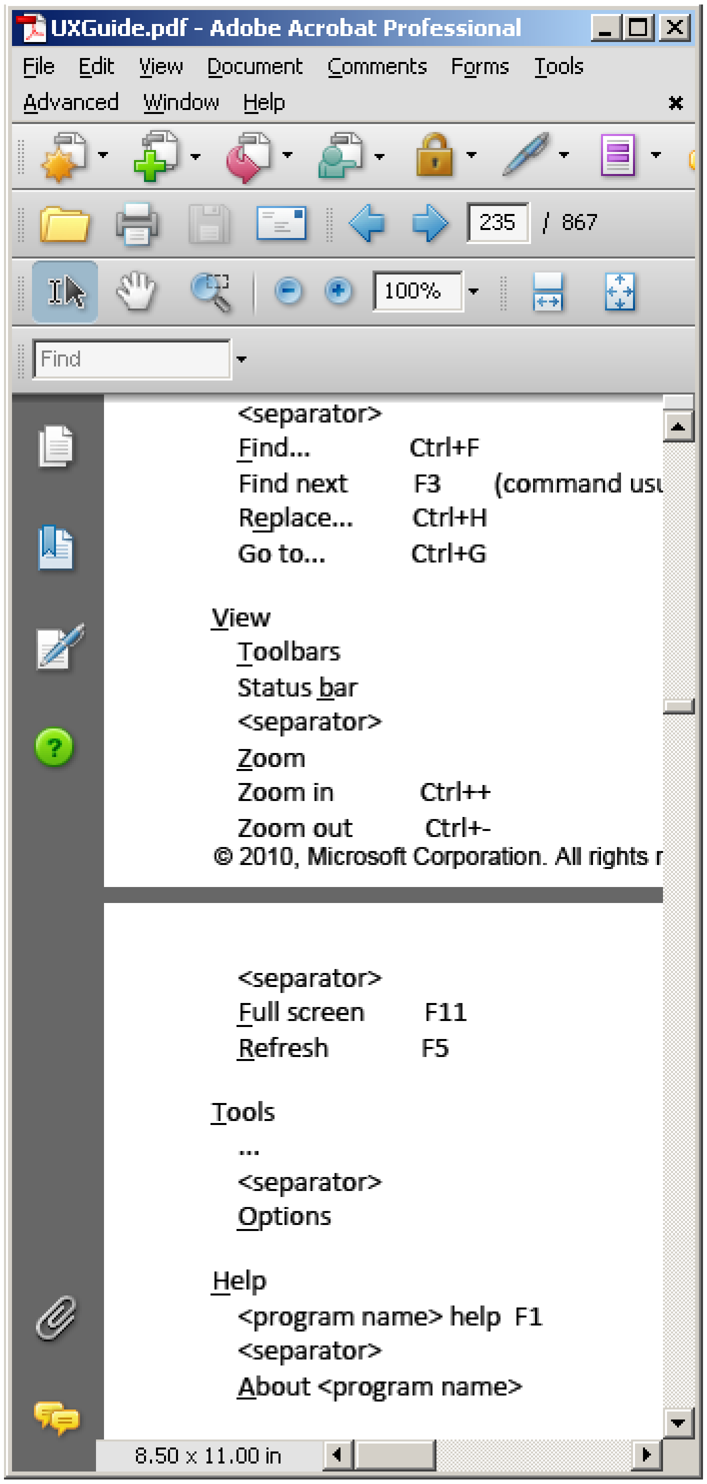

UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
234 / 867
100%
Find
8.50 x 11.00 in

<separator>		
Select all	Ctrl+A	
<separator>		
Delete	Del	(shortcut usua
<separator>		
Find...	Ctrl+F	
Find next	F3	(command usu
Replace...	Ctrl+H	
Go to...	Ctrl+G	
View		
Toolbars		
Status bar		
<separator>		
Zoom		
Zoom in	Ctrl++	
Zoom out	Ctrl+-	

© 2010, Microsoft Corporation. All rights r

<separator>	
Full screen	F11
Refresh	F5
Tools	
...	
<separator>	
Options	
Help	

Figure 3-15: Menu order with separators

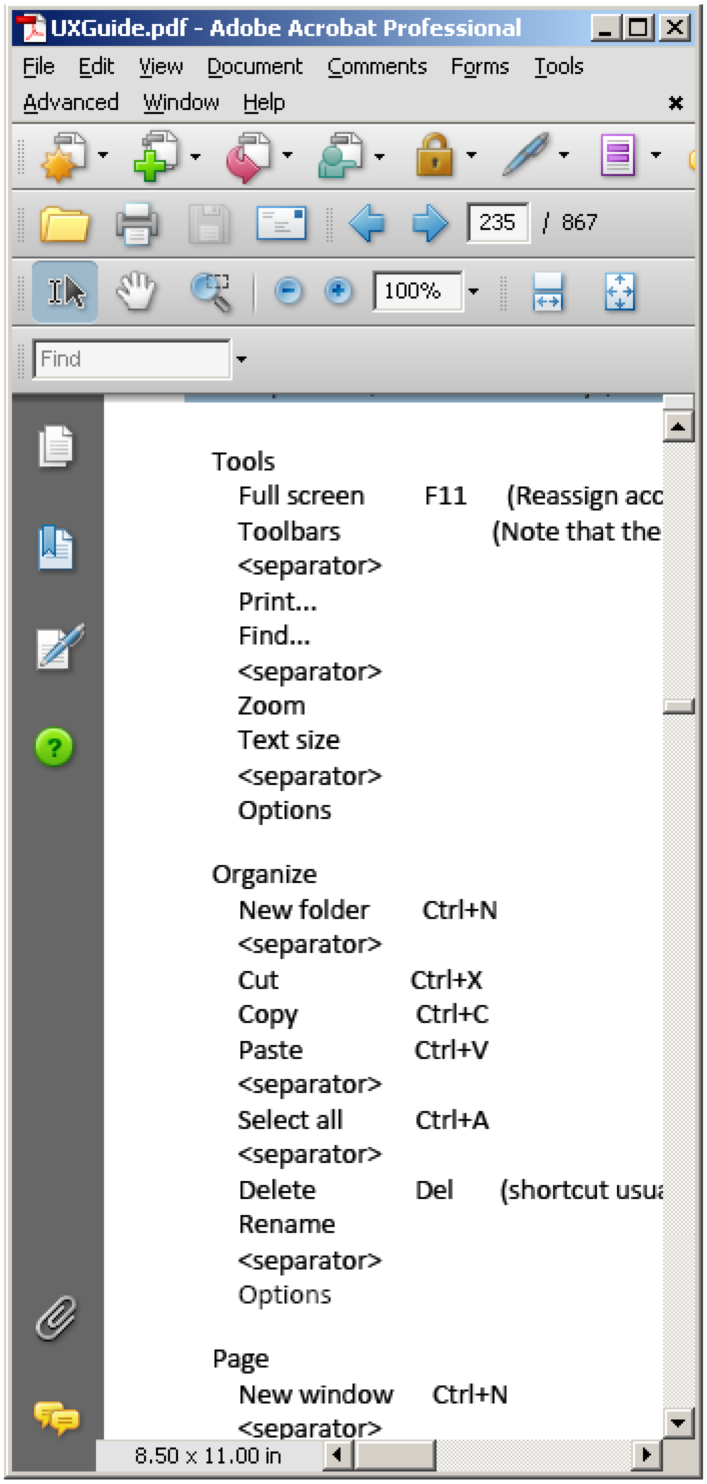

UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
235 / 867
100%
Find
8.50 x 11.00 in

```
    <separator>
    Find...        Ctrl+F
    Find next      F3      (command usu
    Replace...     Ctrl+H
    Go to...       Ctrl+G

View
    Toolbars
    Status bar
    <separator>
    Zoom
    Zoom in          Ctrl++
    Zoom out         Ctrl+-
© 2010, Microsoft Corporation. All rights r

    <separator>
    Full screen      F11
    Refresh          F5

Tools
    ...
    <separator>
    Options

Help
    <program name> help  F1
    <separator>
    About <program name>
```

Figure 3-16: Menu access and shortcut keys

3.24.4.2 Standard Toolbar Menu Buttons

The standard toolbar menu buttons are as follows. This list shows the menu category and item labels, their order with separators, their shortcut keys, and their ellipses.

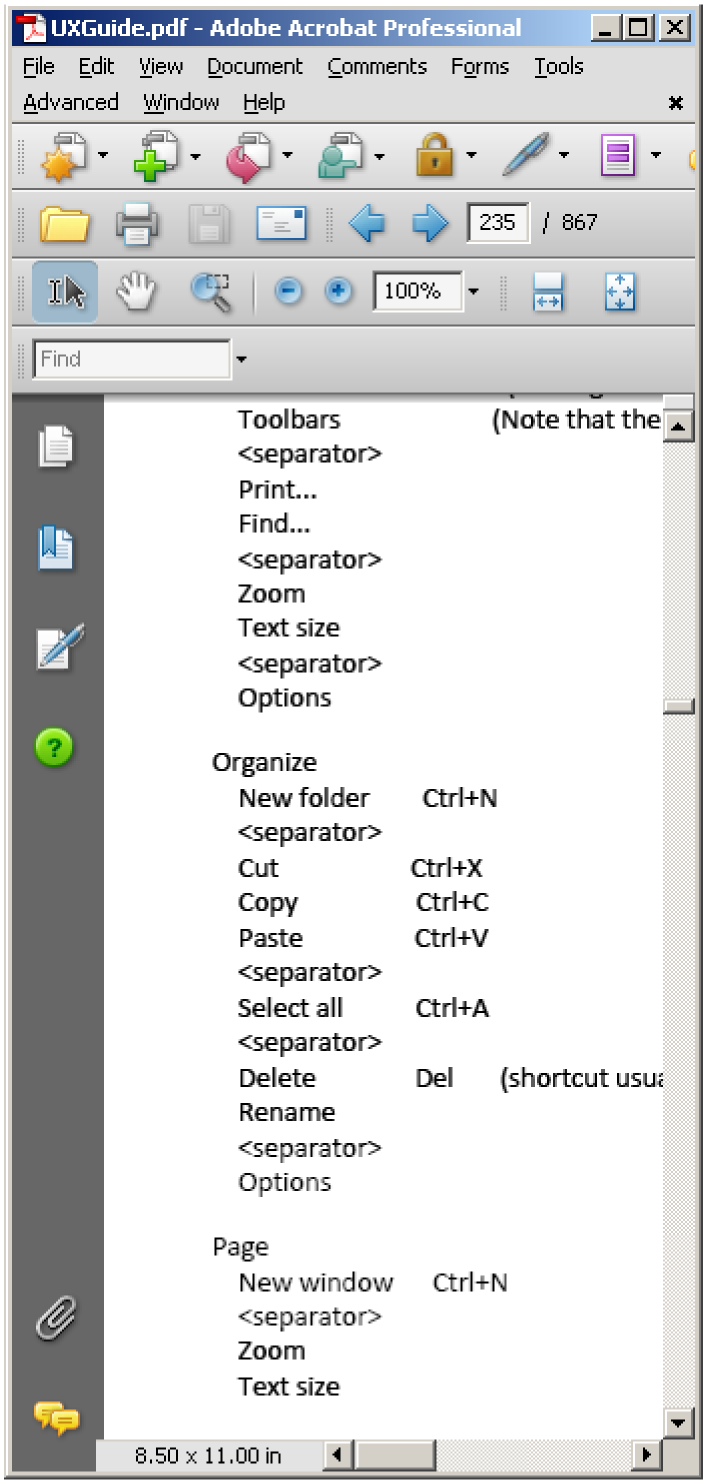

UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
235 / 867
100%
Find
8.50 x 11.00 in

Tools
  Full screen F11 (Reassign acc
  Toolbars (Note that the
  <separator>
  Print...
  Find...
  <separator>
  Zoom
  Text size
  <separator>
  Options

Organize
  New folder Ctrl+N
  <separator>
  Cut Ctrl+X
  Copy Ctrl+C
  Paste Ctrl+V
  <separator>
  Select all Ctrl+A
  <separator>
  Delete Del (shortcut usu
  Rename
  <separator>
  Options

Page
  New window Ctrl+N
  <separator>

Figure 3-17: Standard toolbar menu buttons


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
235 / 867
100%
Find
8.50 x 11.00 in

Toolbars (Note that the
<separator>
Print...
Find...
<separator>
Zoom
Text size
<separator>
Options

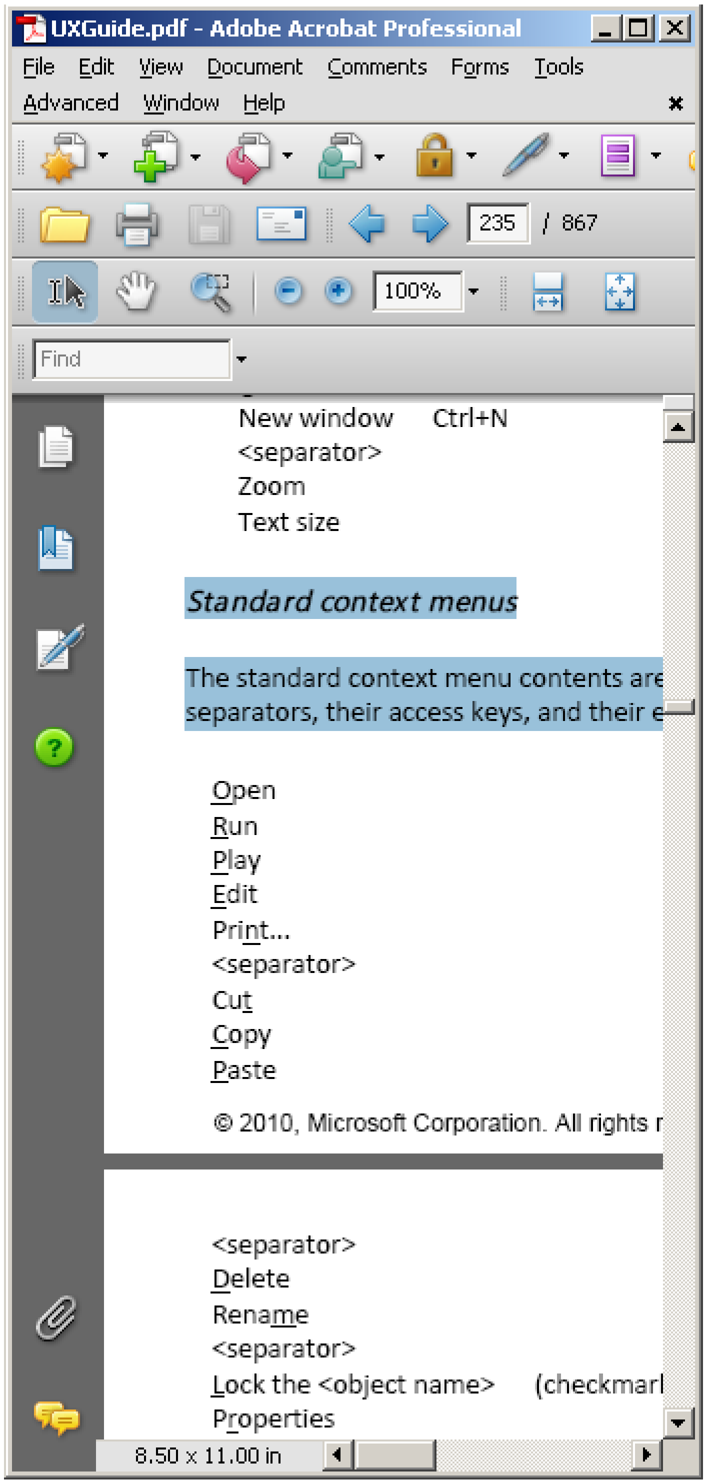
Organize
New folder Ctrl+N
<separator>
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
<separator>
Select all Ctrl+A
<separator>
Delete Del (shortcut usu
Rename
<separator>
Options

Page
New window Ctrl+N
<separator>
Zoom
Text size

Figure 3-18: Menu order with separators

3.24.4.3 Standard Context Menus

The standard context menu contents are as follows. This list shows the menu item labels, their order with separators, their access keys, and their ellipses. Context menus don't show shortcut keys.


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
235 / 867
100%
Find
8.50 x 11.00 in

New window Ctrl+N
<separator>
Zoom
Text size

Standard context menus

The standard context menu contents are
separators, their access keys, and their e

Open
Run
Play
Edit
Print...
<separator>
Cut
Copy
Paste

© 2010, Microsoft Corporation. All rights r

<separator>
Delete
Rename
<separator>
Lock the <object name> (checkmar
Properties

Figure 3-19: Context menu contents

3.24.5 Using Ellipses

While menu commands are used for immediate actions, more information might be needed to perform the action. Indicate a command that needs additional information (including a confirmation) by adding an ellipsis at the end of the label.

In case of ambiguity (for example, the command label lacks a verb), decide based on the most likely user action. If simply viewing the window is a common action, don't use an ellipsis. This doesn't mean you should use an ellipsis whenever an action displays another window—only when additional information is required to perform the action. For example, the commands About, Advanced, Help, Options, Properties, and Settings must display another window when clicked, but don't require additional information from the user. Therefore they don't need ellipses.

In case of ambiguity (for example, the command label lacks a verb), decide based on the most likely user action. If simply viewing the window is a common action, don't use an ellipsis.

Note: When determining if a menu command needs an ellipsis, don't use the need to elevate privileges as a factor. Elevation isn't information needed to perform a command (rather, it's for permission) and the need to elevate is indicated with the security shield.

3.24.6 Labels

Use sentence-style capitalization. Exception: For legacy applications, you may use title-style capitalization if necessary to avoid mixing capitalization styles.

3.24.7 Menu Category Names

Use menu category names that are single word verbs or nouns. A multiple-word label might be confused for two one-word labels.

Prefer verb-based menu names. However, omit the verb if it is Create, Show, View, or Manage. For example, the following menu categories don't have verbs:

- Table
- Tools
- Window

For non-standard category names, use a single, specific word that clearly and accurately describes the menu contents. While the names don't have to be so general that they describe everything in the menu, they should be predictable enough so that users aren't surprised by what they find in the menu.

3.24.8 Menu Item Names

- Use menu item names that start with a verb, noun, or noun phrase.
- Prefer verb-based menu names. However, omit the verb if: The verb is Create, Show, View, or Manage. For example, the following commands don't have verbs:
  - About
  - Advanced
  - Full screen
  - New
  - Options
  - Properties
- **The verb is the same as the menu category name to avoid repetition.** For example, in the Insert menu category, use Text, Table, and Picture instead of Insert text, Insert table, and Insert picture.
- **Use specific verbs.** Avoid generic, unhelpful verbs, such as Change and Manage.
- **Use singular nouns for commands that apply to a single object**, otherwise use plural nouns.
- **Use modifiers as necessary to distinguish between similar commands.** Examples: Insert row above, Insert row below.
- **For pairs of complementary commands, choose clearly complementary names.** Examples: Add, Remove; Show, Hide; Insert, Delete.
- **Choose menu item names based on user goals and tasks, not on technology.**
- Use the following menu item names for the stated purpose:
  - **Options** To display program options.
  - **Customize** To display the program options specifically related to mechanical UI configuration.
  - **Personalize** To display a summary of commonly used **personalization** settings.
  - **Preferences** Don't use. Use Options instead.
  - **Properties** To display an object's property window.
  - **Settings** Don't use as a menu label. Use Options instead.

3.24.9 Submenu Names

Menu items that display submenus never have an ellipsis on their label. The submenu arrow indicates that another selection is required.

3.25 Toolbars (Page 245)

3.25.1 Guidelines

3.25.1.1 Presentation

- **Choose a suitable toolbar style based on the number of commands and their usage.** See the previous **toolbar style** table for guidance on how to choose. Avoid using a toolbar configuration that takes too much space from the program work area.
- **Place toolbars just above the content area,** below the menu bar and address bar, if present.

  If space is at a premium, save space by:

  - Omitting the labels of well-known icons and less frequently used commands.
  - Using only a partial toolbar instead of the entire window width.
  - Consolidating related commands with a **menu button** or **split button**.
  - Using an **overflow chevron** to reveal less frequently used commands.
  - Displaying commands only when they apply to the current context.
- **For the unlabeled icons toolbar pattern, use a default configuration with no more than two rows of toolbars.** If more than two rows might be useful, make the toolbars **customizable**. Starting with more than two rows can overwhelm users and take too much space from the program work area.
- **Disable individual toolbar buttons that don't apply to the current context,** instead of removing them. Doing so makes toolbar contents stable and easier to find.
- **Disable individual toolbar buttons if clicking on them would directly result in an error.** Doing so is necessary to maintain a **direct feel**.
- **For the unlabeled icons toolbar pattern, remove entire toolbars if they don't apply to the current context.** Display them only in the applicable modes.
- **Display toolbar buttons left aligned.** The Help icon, if present, is right aligned.
- **Exception:** Windows 7-style toolbars left align program specific commands, but right align standard, well-known commands such as Options, View, and Help.
- **Don't change toolbar button labels dynamically.** Doing so is confusing and unexpected. However, you can change the icon to reflect the current state.

3.25.2 Controls And Commands

- **Prefer the most frequently used commands. For primary toolbars, provide comprehensive commands.** Primary toolbars don't have to be as comprehensive as menu bars, but they have to provide all the commands that aren't readily discoverable elsewhere. Primary toolbars don't need to have commands for:
  - Commands that are directly on the UI itself.
  - Commands typically accessed through context menus.
  - Standard, well-known commands like Cut, Copy, and Paste.
- **For supplemental toolbars, provide commands that are used the most frequently.** Menu bar commands are a superset of the toolbar commands, so you don't have to provide everything. Focus on quick, convenient command access and skip the rest.
- **Prefer direct controls.** Use toolbar buttons in the following order of preference:
  - **Icon button.** Direct and takes minimal space.
  - **Labeled icon button.** Direct, but takes more space.
  - **Split button.** Direct for the most common command, but handles command variations.
  - **Menu button.** Indirect, but presents many commands.
- **Prefer immediate commands.** For commands that can either be immediate or have additional input for flexibility: For primary toolbars, use the flexible versions of commands, (such as Print...).
- For supplemental toolbars, use the immediate versions in the toolbar (such as Print) and use flexible versions in the menu bar (such as Print...).
- **Provide labels for frequently used commands,** especially if their icons aren't well-known icons.
- Don't put commands in toolbar menus that are also directly on the toolbar.

3.25.3 Organization And Order

- Organize the commands within a toolbar into related groups.
- Place the most frequently used groups first. Within a group, put the commands in their logical order. Overall, the commands should have a logical flow to make them easy to find, while still having the most frequently used commands appear first. Doing so is most efficient, especially if there is overflow.
- Use group dividers only if the commands across groups are weakly coupled. Doing so makes the groupings obvious and the commands easier to find.
  - **Avoid placing destructive commands next to frequently used commands.** Use either order or grouping to get separation. Also, consider not placing destructive commands in the toolbar, but only in the menu bar or context menus instead.

  - **Use the overflow chevron to indicate that not all commands can be displayed.** But use overflow only if there isn't sufficient room to display all the commands.
  - **Make sure that the most frequently used commands are directly accessible from the toolbar (that is, not in overflow) in small window sizes.** If necessary, reorder the commands, move less frequently used commands to menu buttons or split buttons, or even remove them completely from the toolbar. If this remains a problem, reconsider your choice of **toolbar style**.

3.25.4 Hiding Menu Bars

- Hide the menu bar by default if your toolbar design makes having a menu bar redundant.
- Hide the menu bar instead of removing it completely, because menu bars are more accessible for keyboard users.
- To restore the menu bar, provide a Menu bar checkmark option in the View (for primary toolbars) or Tools (for secondary toolbars) menu category. For more information, see **Standard menu and split buttons**.
- Display the menu bar when users press the Alt key, and set input focus on the first menu category.

3.25.5 Interaction

- On hover, display the button **affordance** to indicate that the icon is clickable. After the tooltip timeout, display the tooltip or infotip.
  - On left single-click: For **command buttons**, interact with the control as normal.
  - For **mode buttons**, display the control to reflect the currently selected mode. If the mode affects the behavior of mouse interaction, also change the pointer.
    - For **property buttons** and **drop-down lists**, display the control to reflect the state of the currently selected objects, if any. On interaction, update the control's state and apply the change to the selected objects. If nothing is selected, do nothing.
    - On left double-click, perform the same action as a left single-click. **Exception:** On rare occasions, a toolbar command can be used more efficiently modally. In such cases, use double-click to toggle the mode.
    - On right-click: For customizable toolbars, display the context menu for customizing the toolbar. Display the menu on right-click on mouse down, not mouse up.
    - For other toolbars, do nothing.

3.25.6 Icons

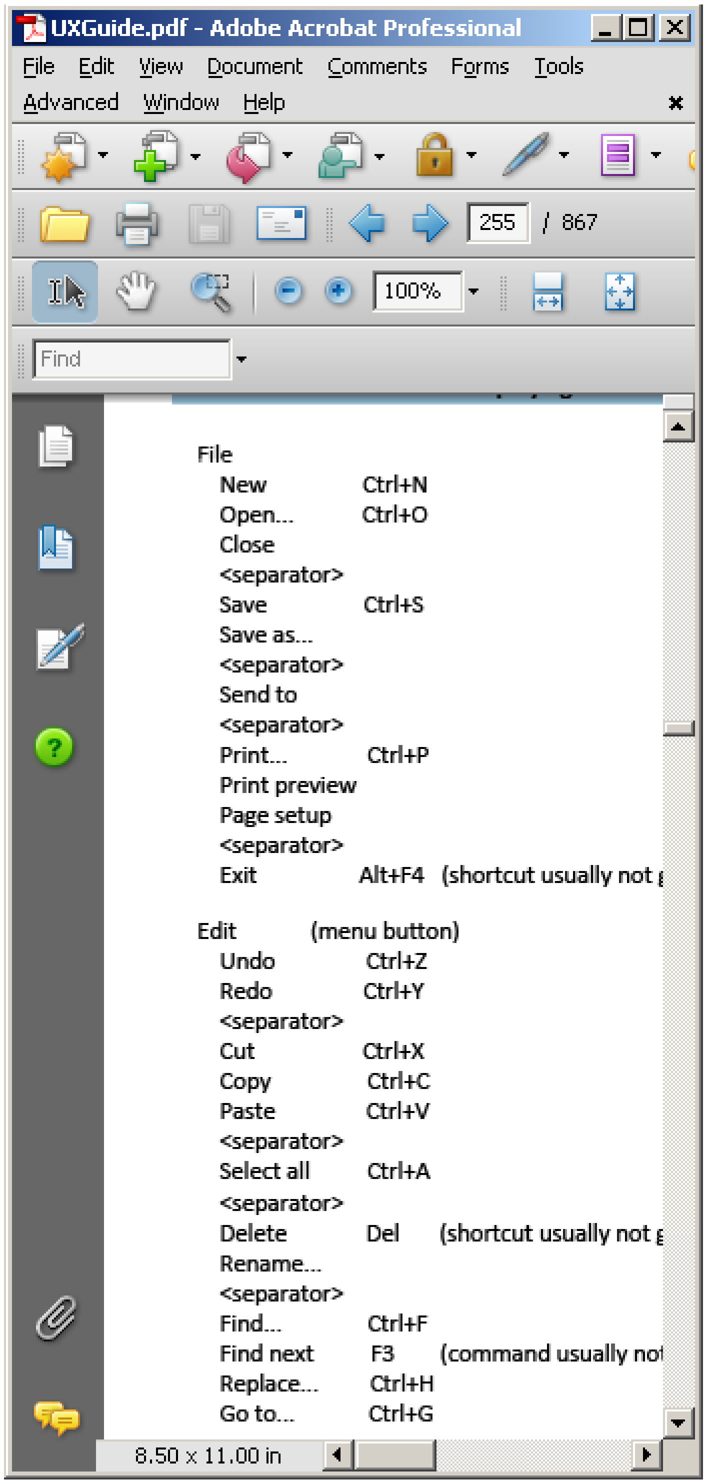
- Provide icons for all toolbar controls except drop-down lists.
- Exception: Windows 7-style toolbars use icons only for commands whose icons are well known; otherwise they use text labels without icons. Doing so improves the clarity of the labels, but requires more space.
- Make sure toolbar icons are clearly visible against the toolbar background color. Always evaluate toolbar icons in context and in high-contrast mode.
- Choose icon designs that clearly communicate their purpose, especially for the most frequently used commands. Well-designed toolbars need icons that are self-explanatory because users can't find commands efficiently using their tooltips. However, toolbars still work well if icons for a few less frequently used commands aren't self-explanatory.
- Choose icons that are recognizable and distinguishable, especially for the most frequently used commands. Make sure the icons have distinctive shapes and colors. Doing so helps users find the commands quickly even if they don't remember the icon symbol.
- Make sure toolbar icons conform to the Aero-style icon guidelines.

3.25.7 Standard Menu And Split Buttons

- If you are using menu buttons and split buttons in a toolbar, try to use the following standard menu structures and their relevant commands whenever possible. Unlike menu bars, toolbar commands don't take access keys.

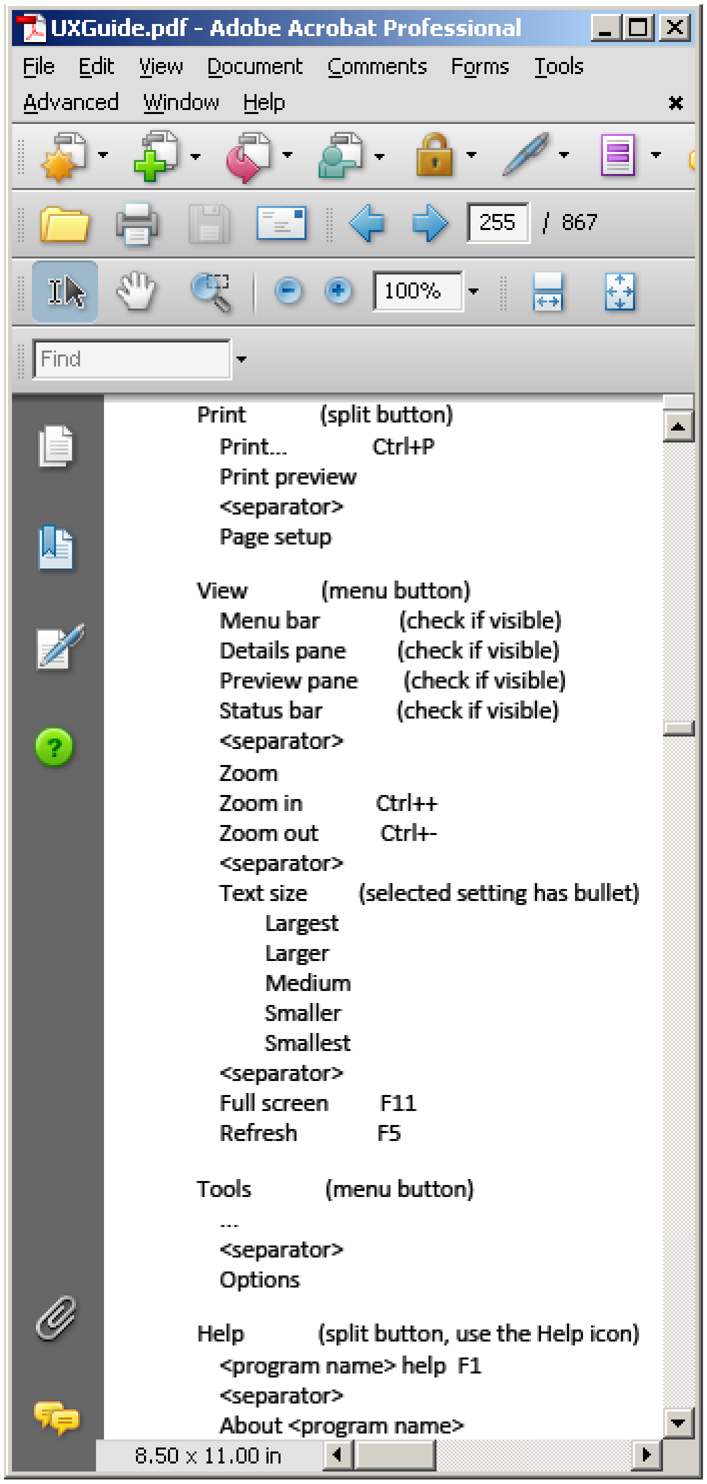
3.25.8 Primary Toolbars

These commands mirror the commands found in standard menu bars, so they should be used only for primary toolbars. This list shows the button labels (and type) with their order and separators, shortcut keys, and ellipses. Note that the command for displaying and hiding the menu bar is in the View menu.


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
255 / 867
100%
Find
8.50 x 11.00 in

File			
	New	Ctrl+N	
	Open...	Ctrl+O	
	Close		
	<separator>		
	Save	Ctrl+S	
	Save as...		
	<separator>		
	Send to		
	<separator>		
	Print...	Ctrl+P	
	Print preview		
	Page setup		
	<separator>		
	Exit	Alt+F4	(shortcut usually not
Edit	(menu button)		
	Undo	Ctrl+Z	
	Redo	Ctrl+Y	
	<separator>		
	Cut	Ctrl+X	
	Copy	Ctrl+C	
	Paste	Ctrl+V	
	<separator>		
	Select all	Ctrl+A	
	<separator>		
	Delete	Del	(shortcut usually not
	Rename...		
	<separator>		
	Find...	Ctrl+F	
	Find next	F3	(command usually not
	Replace...	Ctrl+H	
	Go to...	Ctrl+G	

Figure 3-20: Primary toolbar

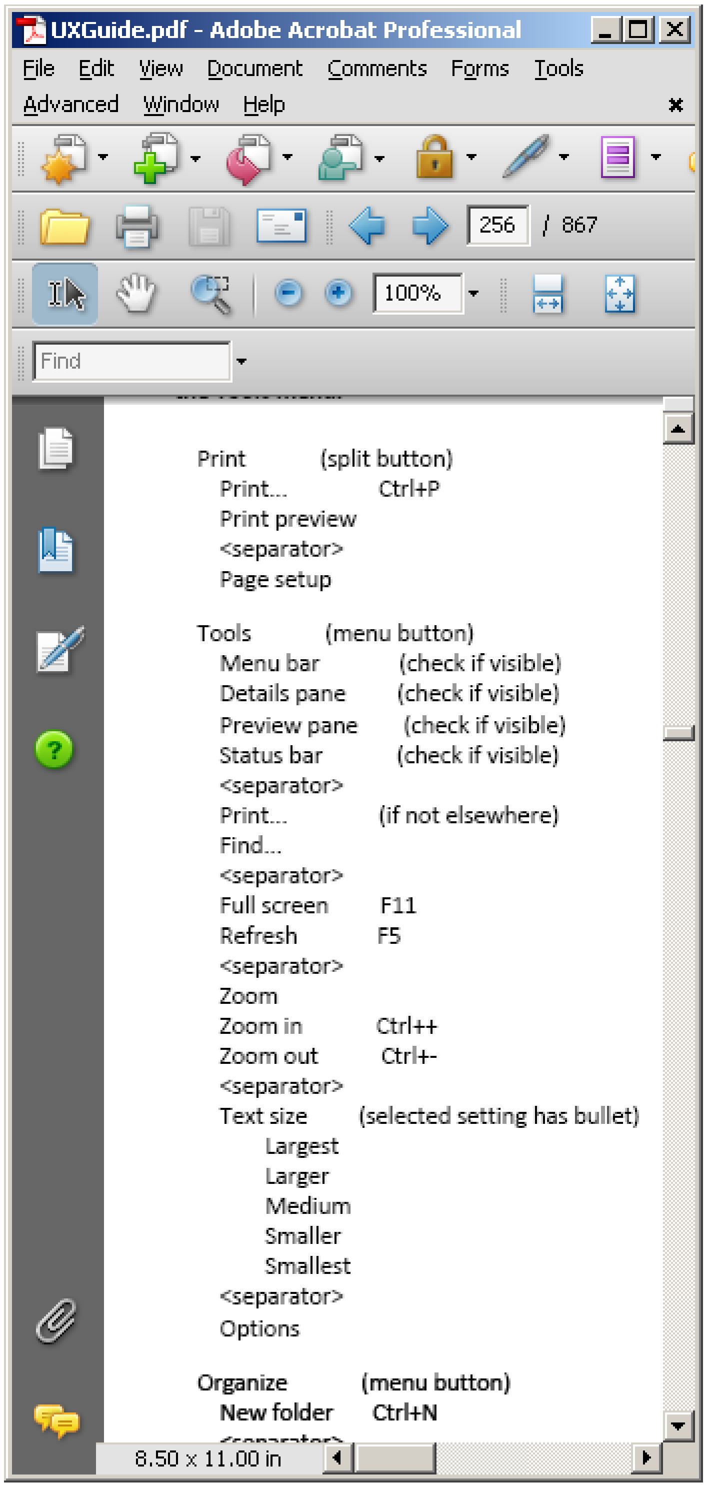

UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
255 / 867
100%
Find
8.50 x 11.00 in

```
Print          (split button)
    Print...          Ctrl+P
    Print preview
    <separator>
    Page setup

View           (menu button)
    Menu bar          (check if visible)
    Details pane      (check if visible)
    Preview pane      (check if visible)
    Status bar        (check if visible)
    <separator>
    Zoom
    Zoom in           Ctrl++
    Zoom out          Ctrl+-
    <separator>
    Text size         (selected setting has bullet)
        Largest
        Larger
        Medium
        Smaller
        Smallest
    <separator>
    Full screen       F11
    Refresh           F5

Tools          (menu button)
    ...
    <separator>
    Options

Help           (split button, use the Help icon)
    <program name> help  F1
    <separator>
    About <program name>
```

Figure 3-21: Menu order and separtors

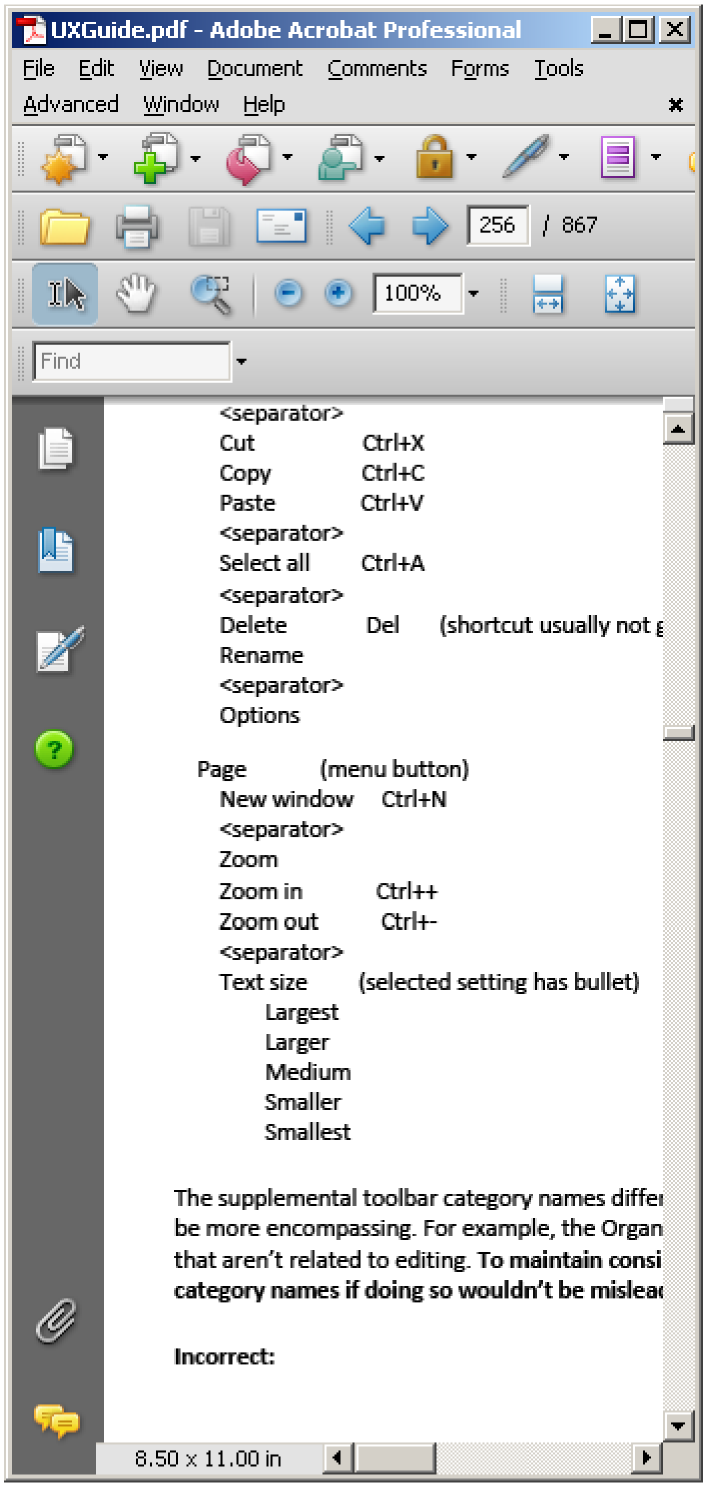
3.25.9 Supplemental Toolbars

These commands supplement standard menu bars. This list shows the button labels (and type) with their order and separators, shortcut keys, and ellipses. Note that the command for displaying and hiding the menu bar is in the Tools menu.


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
256 / 867
100%
Find
8.50 x 11.00 in

```
Print          (split button)
   Print...          Ctrl+P
   Print preview
   <separator>
   Page setup

Tools          (menu button)
   Menu bar          (check if visible)
   Details pane      (check if visible)
   Preview pane      (check if visible)
   Status bar        (check if visible)
   <separator>
   Print...          (if not elsewhere)
   Find...
   <separator>
   Full screen       F11
   Refresh           F5
   <separator>
   Zoom
   Zoom in           Ctrl++
   Zoom out          Ctrl+-
   <separator>
   Text size      (selected setting has bullet)
         Largest
         Larger
         Medium
         Smaller
         Smallest
   <separator>
   Options

Organize          (menu button)
   New folder        Ctrl+N
```

Figure 3-22: Supplemental toolbar


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools
Advanced Window Help
256 / 867
100%
Find
8.50 x 11.00 in

```
    <separator>
    Cut            Ctrl+X
    Copy           Ctrl+C
    Paste          Ctrl+V
    <separator>
    Select all     Ctrl+A
    <separator>
    Delete          Del     (shortcut usually not g
    Rename
    <separator>
    Options

  Page          (menu button)
    New window     Ctrl+N
    <separator>
    Zoom
    Zoom in         Ctrl++
    Zoom out        Ctrl+-
    <separator>
    Text size      (selected setting has bullet)
         Largest
         Larger
         Medium
         Smaller
         Smallest
```

The supplemental toolbar category names differ
be more encompassing. For example, the Organ
that aren't related to editing. **To maintain consi**
category names if doing so wouldn't be mislea

Incorrect:

Figure 3-23: Supplemental toolbar order and separators

The supplemental toolbar category names differ from the standard menu category names because they need to be more encompassing. For example, the Organize category is used instead of Edit because it contains commands that aren't related to editing. To maintain consistency between menu bars and toolbars, use the standard menu category names if doing so wouldn't be misleading.

3.25.10 Palette Windows

- Palette windows use shorter title bars to minimize their screen space. Put a Close button on the title bar.
- Set the title bar text to the command that displayed the palette window.
- Use sentence-style capitalization without ending punctuation.
- Provide a context menu for window management commands. Display this context menu when users right-click on the title bar.
  - **When possible and useful, make palette windows resizable.** Indicate that the window is resizable, using resize pointers when over the window frame.
  - **When a palette window is redisplayed, display it using the same state as last accessed.** When closing, save the window size and location. When redisplaying, restore the saved window size and location. Also, consider making these attributes persistent across program instances on a per user basis.

3.25.11 Customization

- **Provide customization for toolbars consisting of two or more rows.** Only the **unlabeled icons** style needs customization. Simple toolbars with few commands don't need customization.
- **Provide a good default configuration.** Users shouldn't have to customize their toolbars for common scenarios. Don't depend upon users customizing their way out of a bad initial configuration. Assume that most users won't customize their toolbars.
- Provide a context menu with the following commands:
  - A check box list to display the available toolbars
  - Lock/Unlock toolbars
  - Customize...
  - Lock customizable toolbars by default, to prevent accidental changes.
  - For the Customize command, display an options dialog box that provides the ability to choose which toolbars are displayed and the commands on each toolbar.
- Provide a Reset command to return to the original toolbar configuration in the Customize options dialog box.

- Provide the ability to customize the toolbars using drag-and-drop in the following ways: Set toolbar order and positions.
- Set toolbar lengths, displaying any toolbars that are too small to display their contents with an overflow chevron.
- If supported, undock toolbars to become palette windows and vice versa.
- When the Customize options dialog box is displayed:
    - Set the toolbar contents.
    - Set the order of the toolbar contents. Doing so allows users to make changes more directly and efficiently.
    - Save all toolbar customizations, on a per-user basis.

3.25.12 Using Ellipses

- Use an ellipsis to indicate that a command requires more information before it can take effect. Put the ellipsis at the end of the tooltip and label, if there is one.
- If a command cannot take effect immediately, however, no ellipsis is required.

Because toolbars are constantly displayed, and space is at a premium, ellipses should be used infrequently.

Note:	For menus displayed by a toolbar, apply the **menu ellipses guidelines**.

Recommended sizing and spacing

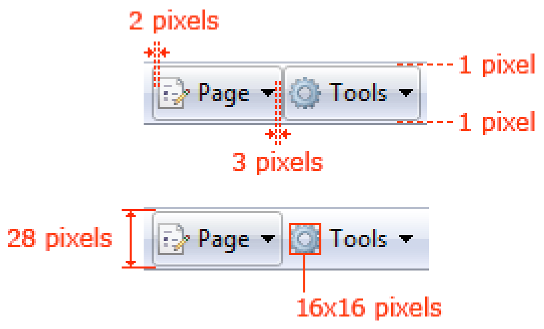


Figure 3-24: Recommended sizing and spacing of toolbar customizations

3.26 Labels (Page 259)

3.26.1 General

Use sentence-style capitalization. Exception: For legacy applications, you may use title-style capitalization if necessary to avoid mixing capitalization styles.

3.26.2 Unlabeled Icon Buttons

Use a tooltip to label the command. For the tooltip text, use what the label would be if the button were labeled, but include the shortcut key if there is one.

3.26.3 Labeled Icon Buttons

- Use a concise label. Use a single word if possible, four words maximum.
- Place the label to the right of the icon.
- Use an infotip to describe the command. Because the buttons are labeled, using a tooltip instead of an infotip would be redundant.

3.26.4 Drop-Down Lists

- If the list always has a value, use the current value as the label.
- If an editable drop-down list doesn't have a value, use a prompt

3.26.5 Menu Buttons and Split Buttons

- Prefer verb-based menu button names. However, omit the verb if it is Create, Show, View, or Manage. For example, Tools and Page menu buttons don't have verbs.
- Use a single, specific word that clearly and accurately describes the menu contents. While the names don't have to be so general that they describe everything in the menu, they should be predictable enough so that users aren't surprised by what they find in the menu.
- While not required, provide infotip descriptions if they are helpful.

3.26.6 Menu Items

- Use menu item names that start with a verb, noun, or noun phrase.
- Prefer verb-based menu names. However, omit the verb if it is Create, Show, View, or Manage. For example, the following commands don't use verbs:
  - About
  - Advanced
  - Full screen
  - New
  - Options
  - Properties
- **Use specific verbs.** Avoid generic, unhelpful verbs, such as Change and Manage.

- **Use singular nouns for commands that apply to a single object,** otherwise use plural nouns.
- **For pairs of complementary commands, choose clearly complementary names.** Examples: Add, Remove; Show, Hide; Insert, Delete.
- **Choose menu item names based on user goals and tasks, not on technology.**
- Use the following menu item names for the stated purpose:
  - **Options:** To display program options.
  - **Customize:** To display the program options specifically related to mechanical UI configuration.
  - **Personalize:** To display a summary of commonly used **personalization** settings.
  - **Preferences:** Don't use. Use Options instead.
  - **Properties:** To display an object's property window.
  - **Settings:** Don't use as a menu label. Use Options instead.
- **Menu items that display submenus never have an ellipsis on their label.** The submenu arrow indicates that another selection is required.

3.27 Ribbons (Page 261)

3.27.1 Guidelines

3.27.2 General

- **Don't combine ribbons with menu bars and toolbars within a window.** Ribbons must be used in place of menu bars and toolbars. However, a ribbon may be combined with palette windows and navigation elements, such as Back and Forward buttons and an Address bar.
- **Always combine a ribbon with an Application button and Quick Access Toolbar.**
- **Select the left-most tab (usually Home) when a program is started.** Don't make the last selected tab persist across program instances.
- **Show the ribbon in its normal state (not minimized) when a program is started for the first time.** Users often leave default settings unchanged, so minimizing the ribbon at program start will likely cause all commands to be less efficient. Also, showing the ribbon initially minimized can be disorienting.
- **Make the ribbon state persist across program instances.** For example, if a user minimizes the ribbon, it should be shown minimized the next time the program is run. But again, don't make the last selected tab persist in this way.

3.27.3 Tabs

- **Whenever practical, use standard tabs.** Using standard tabs greatly improves discoverability, especially across programs. See the **standard ribbon tabs** later in this article.
- **Label the first tab Home, if appropriate.** The Home tab should contain the most frequently used commands. If you have frequently used commands that don't fit into the other tabs, the Home tab is the right place for them.
- Add a new tab if: **Its commands are strongly related to specific tasks, and can be accurately described by the tab label.** Adding the tab should help make its commands easy to find, not harder.
- **Its commands are mostly unrelated to tasks on other tabs.** Adding the tab shouldn't require more tab switching during commonly performed tasks.
- **The tab has enough commands to justify having an extra place to look.** Don't have tabs with only a few commands. **Exception:** Consider adding a tab with a few commands if they are strongly related to a specific task and adding the tab greatly simplifies an overly complex Home tab.
- Generally, having fewer tabs is better, so remove tabs that don't help achieve these goals.
- **For the remaining tabs, place the most frequently used tabs first, while maintaining a logical order across the tabs.**
- **Optimize the tab design so that users find commands quickly and confidently.** All other considerations are secondary.
- **Don't provide a Help tab.** Instead, provide assistance using program-wide Help and enhanced tooltips.
- **Use a maximum of seven core tabs.** If there are more than seven, it becomes difficult to determine which tab has a command. While seven core tabs is acceptable for applications with many commands, most programs should aim for four or fewer tabs.

3.27.4 Contextual Tabs

- **Use a contextual tab to display a collection of commands that are relevant only when users select a particular object type.** If there are only a few, frequently used commands, it may be more convenient and more stable to use a regular tab, and simply disable commands when they don't apply.
  - **Include only the commands that are specific to a particular object type.** Don't put commands only on a contextual tab if users might need them without first selecting an object.

- **Include the commands frequently used when working with a particular object type.** Put frequently used general contextual commands on context menus and minitoolbars to avoid tab switching during commonly performed tasks. Alternatively, consider putting general commands redundantly on a contextual tab if doing so avoids frequent tab switching. But don't overdo this—don't try to include every command that a user might need while working with the object.
- **Choose a contextual tab color that is different from the currently displayed contextual tabs.** The same tab set can appear at a later time using a different color in order to achieve this, but try to use consistent color assignments across invocations whenever possible.
  - Select a contextual tab automatically when: **The user inserts an object.** In this case, select the first contextual tab in the set.
  - **The user double-clicks an object.** In this case, select the first contextual tab in the set.
  - **The user selected a contextual tab, clicked off the object, then immediately clicked an object of the same type.** In this case, return to the previously selected contextual tab.
  - Doing so aids discoverability, improves the perception of stability, and reduces the need to switch tabs. However, leave users in control by not automatically selecting contextual tabs in other circumstances.
  - **When removing a contextual tab that is the active tab, make the Home tab or first tab the active tab.** Doing so appears the most stable.

3.28 Modal Tabs (Page 273)

- Use a modal tab to display a collection of commands that apply with a particular *temporary* mode, and none of the core tabs apply. If some of the core tabs apply, use a contextual tab instead, and disable the commands that don't apply. Because modal tabs are very limiting, they should be used only when there isn't a better alternative.
  - **To close a modal tab, put the Close <mode> command as the last command on the tab.** Use the Close icon to make the command easy to find. Give the mode in the command to prevent confusion about what is being closed.
  - **To close a modal tab, also redefine the Close button on the window's title bar to close the mode instead of the program.** User testing has shown that many users expect this behavior.

3.28.1 Standard Ribbon Tabs

Whenever practical, map your program's commands to these standard tabs, given in their standard order of appearance.

3.28.2 Regular Tabs

- **Home.** Contains the most frequently used commands. If used, it is always the first tab.
- **Insert.** Contains commands to insert content and objects into a document. If used, it is always the second tab.
- **Page layout.** Contains commands that affect the page layout, including themes, page setup, page backgrounds, indenting, spacing, and positioning. (Note that the indenting and spacing groups can be on the Home tab instead, if there is enough room there.) If used, it is always the third tab.
- **Review.** Contains commands to add comments, track changes, and compare versions.
- **View.** Contains commands that affect the document view, including view mode, show/hide options, zooming, window management, and macros—the commands traditionally found in the Windows menu category. If used, it is the last regular tab unless the Developer tab is showing.
- **Developer.** Contains commands used only by developers. If used, it is hidden by default and the last regular tab when displayed.

Most programs don’t need the Review and Developer tabs.

3.28.3 Contextual Tabs

- **Format.** Contains commands related to changing the format of the selected object type. Usually applies to part of an object.
- **Design.** Contains commands, often in galleries, to apply styles to the selected object type. Usually applies to the entire object.
- **Layout.** Contains commands to change the structure of a complicated object, such as a table or chart.

If you have contextual commands related to format, design, and layout, but not enough for multiple tabs, just provide a Format tab.

3.28.4 Groups

- **Whenever practical, use standard groups.** Having common commands appear with the same names and similar locations greatly improves discoverability. See the **standard ribbon groups** later in this article.
- Add a new group if: **Its commands are strongly related and can be accurately described by the group label.** Adding the group should help make its commands easy to find, not harder.

- **Its commands have a weaker relationship to the commands in other groups.** While all the commands on a tab should be strongly related, some command relationships are stronger than others.
- **The group has enough commands to justify having an extra place to look.** Aim for 3–5 commands for most groups. Avoid having groups with only 1–2 commands, although having an in-ribbon gallery without any other commands within a group is acceptable. Having many groups with a single command suggests too much structure or lack of command cohesion.
- Don’t over-organize by adding groups where they aren’t needed.
- Consider splitting a group if: The group has commands that greatly benefit from having extra labels. For example, the commands might need clarification or their labels might have repetitive text.
  - Place the most commonly used groups in the most prominent locations, and make sure there is a logical order for the groups across the tab.
  - Optimize the group design so that users find commands quickly and confidently. All other considerations are secondary.
  - Don’t scale groups containing a single button to a pop-up group icon. When scaling down, leave them as a single button.
  - Use a maximum of seven groups. If there are more than seven groups, it becomes more difficult to determine which group has a command.

3.28.5 Standard Ribbon Groups

Whenever practical, map your program’s commands to these standard groups, which are given within their associated tabs in their standard order of appearance.

3.28.5.1 Main Tab

- Clipboard
- Font
- Paragraph
- Editing

3.28.5.2 Insert Tab

- Tables
- Illustrations

3.28.5.3 Page Layout Tab

- Themes
- Page setup

- Arrange

3.28.5.4 Review Tab

- Proofing
- Comments

3.28.5.5 View Tab

- Document views
- Show/hide
- Zoom
- Window

3.28.6 Commands

3.28.6.1 General

Take advantage of the discoverability and scalability of ribbons by exposing all the commonly used commands. When appropriate, move frequently used commands from dialog boxes to the ribbon, especially those that are known to be hard to find. Ideally, users should be able to perform common tasks without using any dialog boxes.

- **Don't use the scalability of ribbons to justify adding unnecessary complexity.** Continue to exercise restraint—don't add commands to a ribbon just because you can. Keep the overall command experience simple. The following are ways to simplify the presentation: **Use context menus and minitoolbars for in-place, contextual commands.**
- **Move (or keep) rarely used commands in dialog boxes.** Use dialog box launchers to access these commands. You can still use dialog boxes with ribbons! Just try to reduce the need for using them during common tasks.
- Eliminate redundant, seldom used features.

3.28.6.2 Presentation

Present each command on only one tab. Avoid multiple paths to the same command—especially if the command requires many clicks to invoke. It may seem like a convenience to find a command through multiple paths. But keep in mind that when users find what they are looking for, they stop looking. It is all too easy for users to assume that the first path they find is the only path—which is a serious problem if that path is inefficient. Exception: Contextual tabs may duplicate a few commands from the Home and Insert tabs if doing so prevents changing tabs for common contextual tasks.

Within a group, put the commands in their logical order, while giving preference to the most frequently used commands. Overall, the commands should have a logical flow to make them easy to find, while still having the most frequently used commands appear first. Generally, commands with 32 × 32 pixel icons appear before commands with 16 × 16 pixel icons to aid scanning across groups.

Avoid placing destructive commands next to frequently used commands. A command is considered destructive if its effect is widespread and either it cannot be easily undone or the effect isn't immediately noticeable.

Use separators to indicate strongly related commands, such as a set of mutually exclusive options.

Consider using toolbar-style groups for sets of strongly related, well-known commands that don't need labels. Doing so allows you to present many commands in a compact space without affecting discoverability and ease of learning. To be so well known, such commands are frequently used, instantly recognized, and therefore tend to be on the Home tab.

- **Use 32 × 32 pixel icons for the most frequently used and important labeled commands.** When scaling a group down, make these commands the last to convert to 16 × 16 pixel icons.
- **Avoid arbitrary command placement.** Think carefully about your tab and group design to ensure that users aren't wasting time inspecting every tab to find the command they want.
- **Avoid marketing-based placement.** Marketing objectives around the promotion of new features tend to change over time. Consider future versions of your product and how much frustration a constantly changing organization will cause.
- **Prefer direct controls.** A command is direct if invoked with a single click (that is, without navigating through menus). However, with the exception of in-ribbon galleries, direct controls don't support Live preview, so the need for Live preview is also a factor. If a command is among a related set of formatting options, and Live preview is important and practical, use Live preview to indicate the effect of the options—especially if users are likely to choose the wrong option otherwise. If the command is used frequently, use an in-ribbon gallery for directness. If the command is used infrequently, use a drop-down gallery. Otherwise, to obtain directness use ribbon controls in the following order of preference (all other considerations being equal):
  - **Command buttons, check boxes, radio buttons, and in-place galleries.** These are always direct.
  - **Split buttons.** Direct for the most common command, but indirect for the command variations.
  - **Menu buttons.** These are indirect, but present many commands that are easy to find.

  - **Text boxes (with spin controls).** Text input generally requires more effort than the other control types.
- If your ribbon consists mostly of menu buttons when displayed at full size, you might as well use a menu bar.
- **Prefer immediate commands.** A command is immediate if it takes effect immediately (that is, without dialog boxes to gather additional input). If a command might require input, consider using a split button, with the immediate command in the button portion, and the commands that require input in the submenu.

3.28.7 Galleries

Use a **gallery** if:

- **There is a well-defined, related set of choices from which users typically choose.** There may be an unbounded number of variations, but the likely selections should be well contained. If the choices aren't strongly related, consider using separate galleries.
- **The choices are best expressed visually, such as formatting features.** Using thumbnails makes it easier to browse, understand, and make choices. While the choices can be labeled, the selection is made visually and text labels shouldn't be required to understand the choices.
- **The choices show the result that is achieved immediately with a single click.** There shouldn't be any follow-up dialog box to further clarify the user's intention, or a set of steps to achieve the indicated result. If users might want to adjust the choice, let them do so afterwards.

Don't use a gallery to display many regular commands within a group.

Use an in-ribbon gallery if:

- **The choices are used frequently.**
- The choices need the space and are worth the space potentially being taken from other commands.

  For typical usage, there is no need to group or filter the presented choices.

- **The choices can be displayed effectively within the height of a ribbon (which is 48 pixels).**
- **Choose the smallest standard gallery thumbnail size that does the job well.**
  - For in-ribbon galleries, use thumbnails of 16 × 16, 48 × 48, or 64 × 48 pixels.
  - For drop-down galleries, use thumbnails of 16 × 16, 32 × 32, 48 × 48, 64 × 48, 72 × 96, 96 × 72, 96 × 96, or 128 × 128 pixels.
- All gallery items should have the same thumbnail size.

- For in-ribbon galleries:
  - **Display a minimum of three choices; more if there is room.** If there isn't sufficient space to display at least three choices in the typical window size, use a drop-down gallery instead.
  - **Expand in-ribbon galleries to take advantage of available space.** Use the additional space to show more items and make them easier to choose with a single click.
- For drop-down galleries:
  - Display the gallery from either a combo box, drop-down list, split button, or menu button.
  - If the user clicks the main window to dismiss the drop-down gallery, just dismiss the gallery without selecting or modifying the contents of the main window.
  - If a gallery has many choices and some choices are rarely used, simplify the default gallery by focusing on the commonly used choices. For the remaining commands, provide an appropriate command at the bottom of the gallery drop-down. If the command shows a list of more variations, name it "More <singular feature name> options..."
  - If the command presents a dialog box that allows users to create their own custom options, name it "Custom <feature name>..."
- Organize the choices into groups, if doing so makes browsing more efficient.
- If a gallery has many items, consider adding a filter to help users find choices more efficiently. To avoid confusion, initially display the gallery unfiltered. However, most galleries shouldn't require a filter because they shouldn't have so many choices, and using groups should be sufficient.

3.28.8 Previews

- Use previews to show the effect of a command without users having to perform it first. By using helpful previews, you can improve the efficiency and ease of learning of your program, and reduce the need for trial-and-error. For the different types of command previews, see Previews in the Design Concepts section of this article.
- For live previews, make sure that the preview can be applied and the current state restored within 500 milliseconds. Doing so requires the ability to apply formatting changes quickly and in a way that is interruptible. Users must be able to evaluate different options rapidly for live previews to have their full benefit.
- Avoid using text in previews. Otherwise, the preview images will have to be localized.

3.28.9 Icons

- **Provide icons for all ribbon controls except drop-down lists, check boxes, and radio buttons.** Most commands will require both 32 × 32 and 16 × 16 pixel icons (only 16 × 16 pixel icons are used by the Quick Access Toolbar). Galleries typically use 16 × 16, 48 × 48, or 64 × 48 pixel icons.
- **Provide unique icons.** Don’t use the same icon for different commands.
- **Make sure ribbon icons are clearly visible against the ribbon background color.** Always evaluate ribbon icons in context and in high-contrast mode.
- **Choose icon designs that clearly communicate their effect,** especially for the most frequently used commands. Well-designed ribbons have self-explanatory icons to help users find and understand commands efficiently.
- **Choose icons that are recognizable and distinguishable,** especially for the most frequently used commands. Make sure the icons have distinctive shapes and colors. Doing so helps users find the commands quickly, even if they don’t remember the icon symbol.
- **If useful, change the icon to reflect the current state.** Doing so is especially useful for split buttons whose default effect can change.
- **Make sure ribbon icons conform to the Aero-style icon guidelines.** However, ribbon icons are shown straight on instead of being shown in perspective.

3.29 Enhanced Tooltips (Page 282)

- **All ribbon commands should have enhanced tooltips** to give the command name, shortcut key, description, and optional supplemental information. Avoid tooltips that simply restate the label.
  - When practical, completely describe the command using a concise description. Link to Help only if further explanation is really necessary.
  - When helpful, illustrate the effect of the command using a preview.

3.29.1 Access Keys And Keytips

Note: Keytips are the mechanism used to display access keys for commands displayed directly on a ribbon. (Access keys for drop-down menu commands are indicated with an underlined character.) They differ from menu access keys in the following ways: Two character access keys can be used. For example, FP can be used to access the Format painter command.

The access key assignments are shown using tips instead of underlines, so the character width and descenders aren’t a factor in making assignments.

- **Assign access keys to all ribbon tabs and commands.** The only possible exception is for commands coming from legacy add-ins.

For the Application button and Quick Access Toolbar:

- Assign F to the Application button. This assignment is used because of the Application button's similarity to the traditional File menu.

For the Quick Access Toolbar and recently used file lists, assign access keys numerically.

For tabs:

- Assign H to Home.
- Starting with the most frequently used tabs, assign the first letter of the label.
- For any tabs that cannot be assigned to the first letter, choose a distinctive consonant or a vowel in the label.
- For programs that used to support menu bars, strive to maintain access key compatibility to the best extent practical. Avoid assigning different meanings to access keys from legacy menu categories. For example, if the legacy menu bar version of a program had an Edit menu, strive to use an E access key to the equivalent tab. If there is no equivalent tab, don't assign an E access key to any tab to prevent confusion.

For ribbon commands, menus, and submenus: Assign unique access key combinations within a tab. You can reuse access key combinations within different tabs.

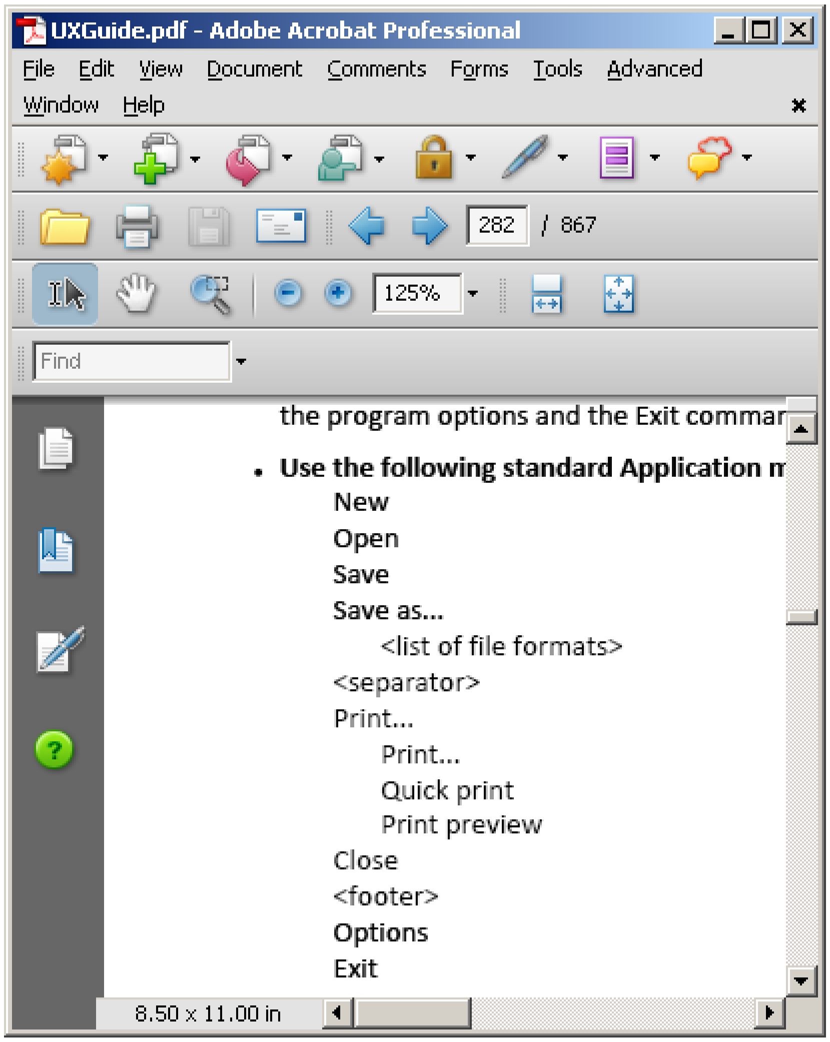
- Whenever possible, assign the standard access keys for commonly used commands. See the **standard access key table**.
  - For other commands: For the most frequently used commands, choose letters at the beginning of the first or second word of the label, preferably the first letter.
  - For less frequently used commands, choose letters that are a distinctive consonant or a vowel in the label, such as "x" in "Exit."
  - For the least frequently used commands and dialog box launchers, use two letters as necessary.
  - For menus and submenus, use a single letter to reduce the number of keystrokes required for the complete command.
- Don't use access keys starting with J, Y, or Z because they are used for contextual tabs, unassigned keytips, and popup groups.
  - **For pop-up groups:** Use a two-letter access key that starts with Z.
  - Starting with the most frequently used groups, assign the second access key letter to the first letter of the label.

– For any remaining groups, choose a distinctive consonant or a vowel in the label.

3.29.2 Application Buttons

- **Use an Application button to present a menu of commands that involve doing something to or with a file.** Examples include commands that traditionally go in the File menu to create, open, and save files, print, and send and publish documents.
- **Always provide an Application button when using a ribbon.** If the program doesn't use files, use the Application button to access the program options and the Exit command. Application buttons always display a command menu—they are never just decorative.

Use the following standard Application menu commands when appropriate:


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools Advanced
Window Help
282 / 867
125%
Find
8.50 x 11.00 in

the program options and the Exit comman

- Use the following standard Application m

  New
  Open
  Save
  Save as...
    <list of file formats>
  <separator>
  Print...
    Print...
    Quick print
    Print preview
  Close
  <footer>
  Options
  Exit

Figure 3-25:Application menu commands

- Reserve commands that belong in the Application menu only for that menu. Don't place them redundantly in any of the tabs.
- For each menu item, provide:
  – A label with the command name.
  – A 32 × 32 pixel icon.
  – A brief description. Make sure the description can be displayed using at most two lines of text.
  – Use tooltips to document the shortcut keys. Unlike normal menus, Application menus don't document the shortcut keys using labels.

3.29.3 Quick Access Toolbars

- Use the Quick Access Toolbar to provide access to frequently used commands. The commands can be from the Application button or the ribbon.
- Always provide a Quick Access Toolbar when using a ribbon. Do so even if the ribbon has a single tab; this provides consistency across programs.
- Prepopulate the Quick Access Toolbar with the frequently used commands in the Application menu. Provide Save and Undo if your program supports them, and Open and Print if supported and frequently used.
- For the Customize Quick Access Toolbar menu, provide up to 12 of the most frequently used immediate commands. Immediate commands don't require additional input before they take effect, and are therefore well-suited for the Quick Access Toolbar. While these can be any immediate commands, prefer those commands that aren't on the Home tab, because users are more likely to choose those.
- For the Customize Quick Access Toolbar menu, if there is a pair of related commands, provide both, regardless of frequency. Common pairs are Open/Close, Back/Forward, and Undo/Redo.
- For the Customize Quick Access Toolbar dialog, provide a way to add any command. Provide a Popular commands filter that displays the most frequently used commands, and select this filter by default.

3.29.4 Dialog Box Launchers

- **Provide a group with a dialog box launcher if there is a related dialog box with infrequently used commands and settings.** The dialog box should contain all the commands in the group, plus others—not a completely different set of commands or the same commands as the group.
- **Don't use a dialog box launcher to perform commands directly.** A dialog box launcher must display a dialog box.

- **Don't use a dialog box launcher to access frequently used commands and settings.** Compared to commands directly on the ribbon, the dialog box commands and settings are relatively undiscoverable.
- **Match the name of the dialog box with the name of the group.** It doesn't have to be an exact match, but the names should be similar enough so that users aren't surprised by the results.
- **Display only the commands and settings that relate to the group.** If the dialog box displays other things, users may conclude that this path to these other commands and settings is the only path.

3.29.5 Labels

3.29.5.1 Tabs

- Label all tabs.
- Whenever practical, use the standard ribbon tabs.
- Prefer concise, single word labels. While multiword labels are acceptable, they take more space and are harder to localize.
  - **Choose meaningful tab names that clearly and accurately describe their content.** The names should be specific, but not overly specific. Tab names should be predictable enough so that users aren't surprised by their content. Note that the Home tab is generically named because it is used for the most frequently used commands.
  - **Choose tab names that reflect their purpose.** Consider the goals or tasks associated with the tab.
  - **Choose tab names that are clearly distinct from all the other tab names.**
  - **Use either nouns or verbs for tabs.** Tab names don't require parallel phrasing, so choose the best label regardless of whether it's a noun or verb.
  - **Don't use gerunds** (names that end in "-ing"). Use the verb from which the gerund is derived instead.
  - **Avoid tab names with the same initial letters, especially adjacent tabs.** When the ribbon is scaled down, these tab names will have the same truncated text.
  - **Prefer singular names.** However, you can use a pural name if the singular name is awkward.
  - **Use title-style capitalization.**
- Don't use ending punctuation.

3.29.6 Contextual Tabs And Tab Sets

- **End contextual tab set labels with "Tools".** Doing so helps identify the purpose of contextual tabs.
- Use title-style capitalization.
- Don't use ending punctuation.

3.30 Groups (Page 285)

- **Label all groups. Exception:** Omit the group label if the group has a single command and the group and command labels would be the same.
- Whenever practical, **use the standard ribbon groups.**
- **Prefer concise, single word labels.** While multiword labels are acceptable, they take more space and are harder to localize.
- **Choose meaningful group names that clearly and accurately describe their content.** The names should be specific, not generic.
- **Choose group names that reflect their purpose.** Consider the goals or tasks associated with the commands in the group.
- **Avoid using gerunds** (names that end in "-ing"). You can use gerunds, however, if using the verb from which the gerund is derived would be confusing. For example, use "Editing" and "Proofing" instead of "Edit" and "Proof."
- **Don't use group names that are the same as tab names.** Using the tab name that the group is on provides no information, and using the name of a different tab is confusing.
- **Prefer singular names.** However, you can use a pural name if the singular name is awkward.
  - Use sentence-style capitalization.
  - Don't use ending punctuation.

3.30.1 Commands

Label all commands. Having explicit text labels helps users find and understand commands. **Exception:** A command can be unlabeled if its icon is extremely well known and space is at a premium. Most likely, unlabeled commands will be on the Home tab. In this case, assign its Name property to an appropriate text label. This enables assistive technology products such as screen readers to provide users with alternative information about the graphic.

- For command buttons, use a concise, self-explanatory label. Use a single word if possible; four words maximum.

- For drop-down lists, if the list always has a value, use the current value as the label.
- If an editable drop-down list doesn't have a value, use a prompt.
  - Drop-down lists that aren't self-explanatory or are infrequently used need an explicit label. Put a colon at the end of the label.
- For text boxes, use an explicit label. Put a colon at the end of the label.
- Use sentence-style capitalization. Doing so is more appropriate for the Windows tone.
  - **Start the label with an imperative verb.** Exceptions: Omit the verb if it's the same as the tab or group name.
  - Omit common verbs like Show, Create, Insert, or Format if the verb is easily inferred from the remaining label.
  - Don't use ending punctuation.
  - **To conserve space, don't put ellipses on ribbon command labels.** However, ellipses are used by commands in the Application button and drop-down menus.

3.30.2 Enhanced Tooltips

- Use the title to give the command name and its shortcut key, if applicable.
- For the title, don't use ending punctuation.
- Start the description with a verb. Use the description to help users determine whether a specific feature is the one they are looking for. The description should be phrased to complete the sentence "This is the right feature to use if you want to...".
- Keep the description short. Get right to the point. Lengthy text discourages reading.
- For split buttons, use a different tooltip to explain the split button menu.
- Use an optional supplemental description to explain how to use the control. This text can include information about the state of the control (including why it is disabled) if the control itself doesn't indicate state. Keep this text short, and use a Help topic for more detailed explanations.
- For the description and supplemental description, use complete sentences with ending punctuation.

3.30.3 Application Buttons

- Use "Quick" to indicate an immediate version of a command.
  - Use an ellipsis to indicate that a command requires more information.

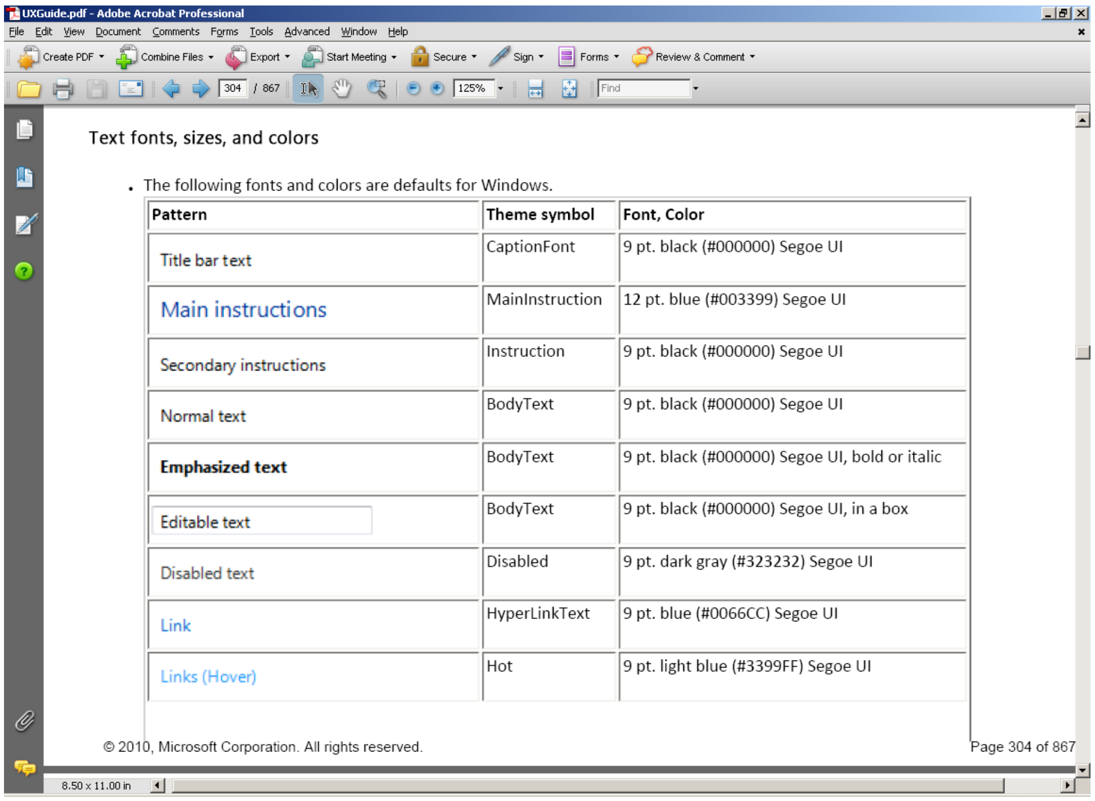
  - Use sentence-style capitalization.

3.31 Text (Page 298)

3.31.1 Guidelines

3.31.1.1 General

- **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.
  - **Avoid large blocks of UI text.** Ways of doing this include chunking text into shorter sentences and paragraphs.
  - When necessary, providing **Help links** to useful, but not essential, information.
  - **Choose object names and labels that clearly communicate and differentiate what the object does.** Users shouldn't have to figure out what the object really means or how it differs from other objects.
    - If you want to make sure that users read specific text related to an action, place it on an interactive control.
    - Use one space between sentences. Not two.



UXGuide.pdf - Adobe Acrobat Professional

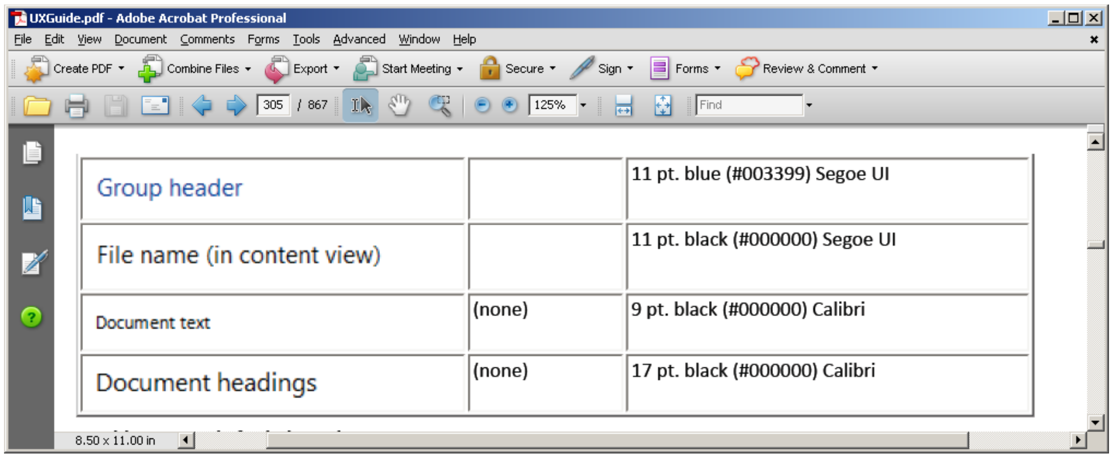
Text fonts, sizes, and colors

- The following fonts and colors are defaults for Windows.

Pattern	Theme symbol	Font, Color
Title bar text	CaptionFont	9 pt. black (#000000) Segoe UI
Main instructions	MainInstruction	12 pt. blue (#003399) Segoe UI
Secondary instructions	Instruction	9 pt. black (#000000) Segoe UI
Normal text	BodyText	9 pt. black (#000000) Segoe UI
Emphasized text	BodyText	9 pt. black (#000000) Segoe UI, bold or italic
Editable text	BodyText	9 pt. black (#000000) Segoe UI, in a box
Disabled text	Disabled	9 pt. dark gray (#323232) Segoe UI
Link	HyperLinkText	9 pt. blue (#0066CC) Segoe UI
Links (Hover)	Hot	9 pt. light blue (#3399FF) Segoe UI

© 2010, Microsoft Corporation. All rights reserved. Page 304 of 867

Figure 3-26: Test fonts, sizes, and colors



UXGuide.pdf - Adobe Acrobat Professional

Group header		11 pt. blue (#003399) Segoe UI
File name (in content view)		11 pt. black (#000000) Segoe UI
Document text	(none)	9 pt. black (#000000) Calibri
Document headings	(none)	17 pt. black (#000000) Calibri

Figure 3-27: Use blue text for main instructions

- **Use blue text only for links and main instructions.**
- **Use green text only for URLs in search results.**

3.31.2 Other Text Characteristics

3.31.2.1 Bold

- **Use bold sparingly to draw attention to text users must read.** For example, users scanning down a list of radio button options may appreciate seeing the labels in bold, to stand out from text that adds supplemental information about each option. Be aware that using too much bold lessens its impact.
- **With labeled data, use bold to emphasize whichever is more important for the data as a whole.** For mostly generic data (where the data has little meaning without its labels, as with numerals or dates), use bold labels and plain data so that users can more easily scan and understand the types of data.
- For mostly self-explanatory data, use plain labels and bold data so that users can focus on the data itself.
- Alternatively, you can use dark gray text to deemphasize less important information instead of using bold to emphasize the more important information.
  - Not all fonts support bold, so it should never be crucial to understanding the text.

3.31.2.2 Italic

- Use to refer to text literally. Don’t use quotation marks for this purpose.
- Use for **prompts** in **text boxes** and **editable drop-down lists**.
- Use sparingly to emphasize specific words to aid in comprehension.
- **Not all fonts support italic, so it should never be crucial to understanding the text.**

3.31.2.3 Bold italic

Don’t use bold italic in UI text.

3.31.2.4 Underline

- Don’t use, except for links.
- Don’t use for emphasis. Use italic instead.

3.31.3 Punctuation

3.31.3.1 Periods

- Don’t place at the end of control labels, main instructions, or Help links.

- Place at the end of supplemental instructions, supplemental explanations, or any other static text that forms a complete sentence.

3.31.3.2 Question Marks

Place at the end of all questions. Unlike periods, question marks are used for all types of text.

3.31.3.3 Exclamation Points

In business applications, avoid. **Exceptions:** Exclamation points are sometimes used in the context of download completion (“Done!”) and to call attention to Web content (“New!”).

3.31.3.4 Commas

In a list of three or more items, always put a comma after the next-to-last item in the list.

3.31.3.5 Colons

- **Use colons at the end of external control labels.** This is particularly important for accessibility because some assistive technologies look for colons to identify control labels.
- Use a colon to introduce a list of items.

3.31.3.6 Ellipses

- **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. For more information, see **Command Buttons**.
- **Data:** Indicate that text is truncated.
- **Labels:** Indicate that a task is in progress (for example, “Searching...”).

3.31.3.7 Quotation Marks And Apostrophes

- To refer to text literally, use italic formatting rather than quotation marks.
- Put window titles and control labels in quotation marks only if required to prevent confusion and you can’t format using bold instead.
- When possible, use curved quotation marks and apostrophes instead of straight ones.
- For quotation marks, prefer double-quotation marks (“ ”); avoid single-quotation marks (‘ ’).

3.31.3.8 Capitalization

- **Use title-style capitalization for titles, 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.
    - 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.
  - **Don't use all capital letters for emphasis.** Studies have shown that this is hard to read, and users tend to regard it as "screaming." For warnings, use a warning icon and a clearly-worded explanation of the situation. There is no need to add, for example, the term WARNING in all capital letters.

3.31.4 Dates And Times

- **Don't hard-code the format of dates and times.** Respect the user's choice of locale and customization options for the date and time formats. The user selects these in the Region and Language control panel item.
- **Use the long date format for scenarios that benefit from having additional information.** Use the short date format for contexts that don't have sufficient space for the long format. While users choose what information they would like to include in the long and short formats, designers choose which format to display in their programs based on the scenario and the context.

3.31.5 Globalization And Localization

- For controls with variable contents (such as list views and tree views), **choose a width appropriate for the longest valid data.**
- **Include space enough in the UI surface for an additional 30%** (up to 200% for shorter text) for any text (but not numbers) that will be localized. Translation from one language to another often changes line length of text.

- Don't compose strings from substrings at run time. Instead, use complete sentences so that there is no ambiguity for the translator.
- **Don't use a subordinate control, the values it contains, or its units label to create a sentence or phrase.** Such a design is not localizable because sentence structure varies with language.
  - Don't make only part of a sentence a link, because when translated, that text might not remain together. Link text should therefore form a complete sentence by itself. **Exception:** Glossary links can be inserted inline, as part of a sentence.

3.31.6 Title Bar Text

- Choose the title bar text based on the type of window: **Top-level, document-centric program windows:** Use a "document name–program name" format. Document names are displayed first to give a document-centric feel.
- **Top-level program windows that are not document-centric:** Display the program name only.
- **Dialog boxes:** Display the command, feature, or program from which the dialog box came. Don't use the title to explain the dialog box's purpose—that's the purpose of the main instructions. For more guidelines, see **Dialog Boxes**.
- **Wizards:** Display the wizard name. Note that the word "wizard" should not be included in wizard names.
- **For top-level program windows, if the title bar caption and icon are displayed prominently near the top of the window, you can hide the title bar caption and icon to avoid redundancy.** However, you still have to set a suitable title internally for use by Windows.
- **For dialog boxes, don't include the words "dialog" or "progress" in the titles.** These concepts are implied and leaving these words off makes the titles easier for users to scan.

3.31.7 Main Instructions

- **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. Incorrect:**
  - **Exceptions:** Error messages, warning messages, and confirmations may use different sentence structures in their main instructions.

  - **Use specific verbs whenever possible.** Specific verbs (examples: connect, save, install) are more meaningful to users than generic ones (examples: configure, manage, set). For control panel pages and wizard pages, if you can't use a specific verb, you may prefer to omit the verb completely.
  - For dialogs, such as error messages and warnings, don't omit the verb.
  - Don't feel obliged to use main instruction text if adding it would only be redundant or obvious from the context of the UI
- **Be concise—use only a single, complete sentence.** Pare the main instruction down to the essential information. If you must explain anything more, consider using a supplemental instruction.
- Use **sentence-style capitalization**.
- **Don't include final periods if the instruction is a statement.** If the instruction is a question, include a final question mark.
- **For progress dialogs, use a gerund phrase briefly explaining the operation in progress,** ending with an ellipsis. Example: "Printing your pictures..."

3.31.8 Supplemental Instructions

When necessary, **use a supplemental instruction to present additional information helpful to understanding or using the window or page,** such as:

- Providing context to explain why the window is being displayed if it is program or system initiated.
- Qualifying information that helps users decide how to act on the main instruction.
- Defining important terminology.
  - **Don't use a supplemental instruction if one isn't necessary.** Prefer to communicate everything with the main instruction if you can do so concisely.
  - **Don't repeat the main instruction with slightly different wording.** Instead, omit the supplemental instruction if there is nothing more to add.
  - Use complete sentences and sentence-style capitalization.

3.31.9 Control Labels

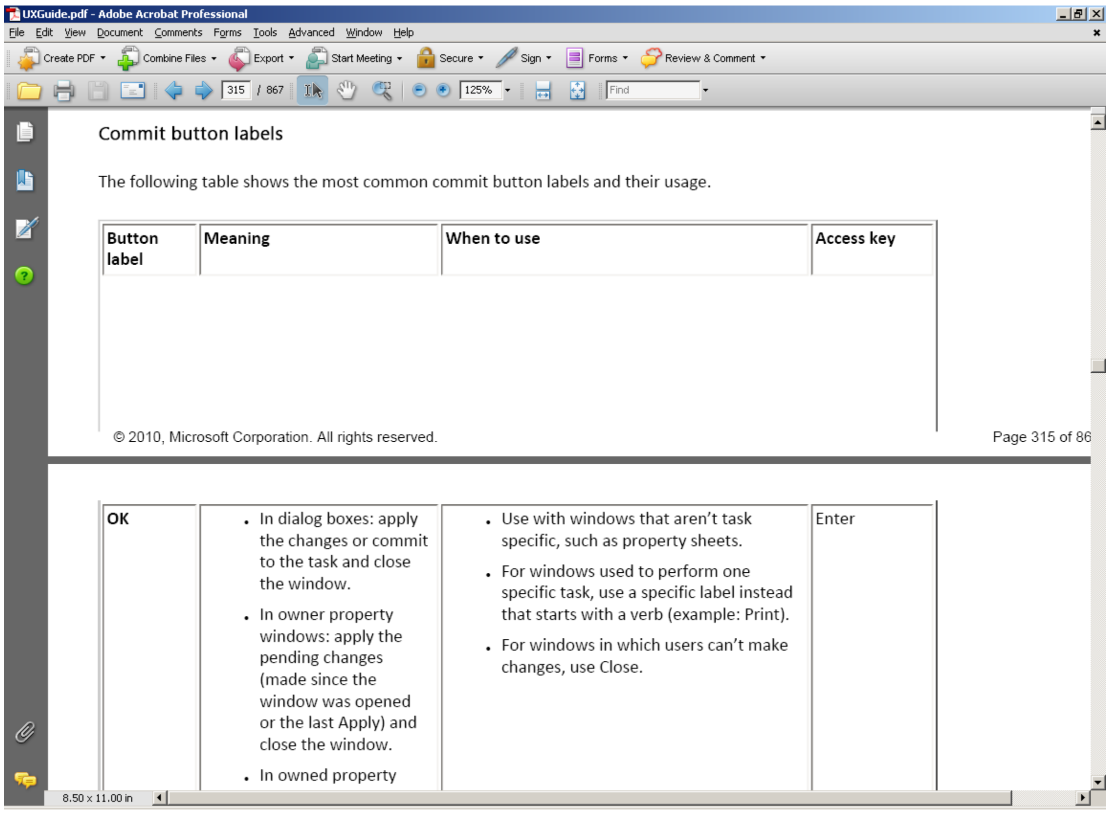
- Label every control or group of controls. Exceptions: Text boxes and drop-down lists can be labeled using prompts.
- Progressive disclosure controls are generally unlabeled.
- Subordinate controls use the label of their associated control. Spin controls are always subordinate controls.
- Omit control labels that restate the main instruction. In this case, the main instruction takes the access key.

- Label placement:
  - Balloons, check boxes, command buttons, group boxes, links, tabs, and tips are labeled directly by the control itself.
  - Drop-down lists, list boxes, list views, progress bars, sliders, text boxes, and tree views are labeled above, flush left, or to the left.
  - Progressive disclosure controls are usually unlabeled. Chevron buttons are labeled to the right.
  - **Assign a unique access key for each interactive control** except for links. **Keep labels brief.** Note, however, that adding a word or two to a label can help clarity, and sometimes eliminates the need for supplemental explanations.
  - **Prefer specific labels over generic ones.** Ideally users shouldn't have to read anything else to understand the label.
- For lists of labels, such as radio buttons, use parallel phrasing, and try to keep the length about the same for all labels.
- For lists of labels, focus the label text on the differences among the options. If all the options have the same introductory text, move that text to the group label.
  - **In general, prefer positive phrasing.** For example, use *do* instead of *do not*, and *notify* instead of *do not notify*. **Exception:** The check box label, "Don't show this message again," is widely used.
  - **Omit instructional verbs that apply to all controls of the given type.** Rather, focus labels on what is unique about the controls. For example, it goes without saying that users need to *type* into a text box control or that users need to *click* a link.
  - In some cases, the following parenthetical annotations to control labels may be helpful: **If an option is optional, consider adding "(optional)" to the label.**
  - **If an option is strongly recommended, add "(recommended)" to the label.** Doing so means the setting is optional, but should be set anyway.
  - **If an option is intended only for advanced users, consider adding "(advanced)" to the label.**
  - You may specify units (seconds, connections, and so on) in parenthesis after the label.

3.32 Supplemental Explanations (Page 314)

- **Use supplemental explanations when controls require more information than can be conveyed by their label.** But don't use a supplemental explanation if one isn't necessary—prefer to communicate everything with the control label if you can do so concisely. Typically, supplemental explanations are used with command links, radio buttons, and check boxes.

– When necessary, use bold in the control labels to make the text easier to scan when there are supplemental explanations.
– Adding a supplemental explanation to one control in a group doesn't mean that you have to provide explanations for all the other controls in the group. Provide the relevant information in the label if you can and use explanations only when necessary. Don't have supplemental explanations that merely restate the label for consistency.
– If a supplemental explanation follows a command link, write the supplemental text in second person.
– Use complete sentences and ending punctuation.


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools Advanced Window Help
Create PDF
Combine Files
Export
Start Meeting
Secure
Sign
Forms
Review & Comment
315 / 867
125%
Find
8.50 x 11.00 in

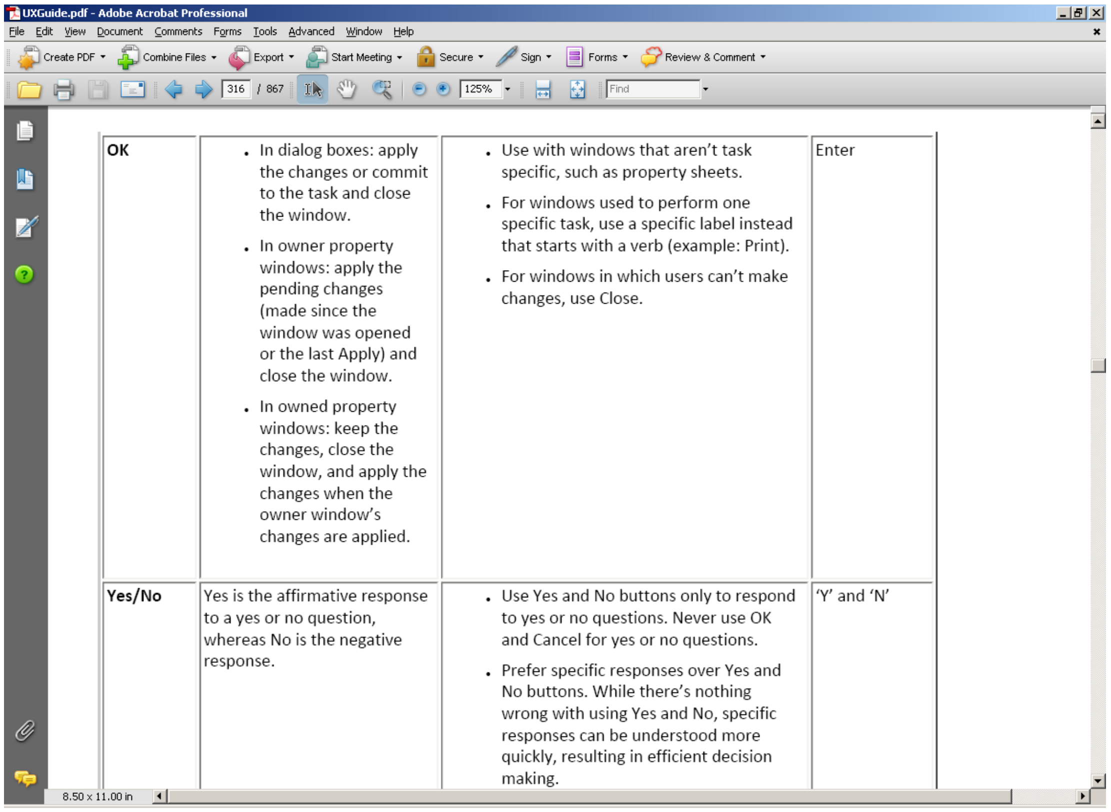
Commit button labels

The following table shows the most common commit button labels and their usage.

Button label	Meaning	When to use	Access key
OK	• In dialog boxes: apply the changes or commit to the task and close the window. • In owner property windows: apply the pending changes (made since the window was opened or the last Apply) and close the window. • In owned property	• Use with windows that aren't task specific, such as property sheets. • For windows used to perform one specific task, use a specific label instead that starts with a verb (example: Print). • For windows in which users can't make changes, use Close.	Enter

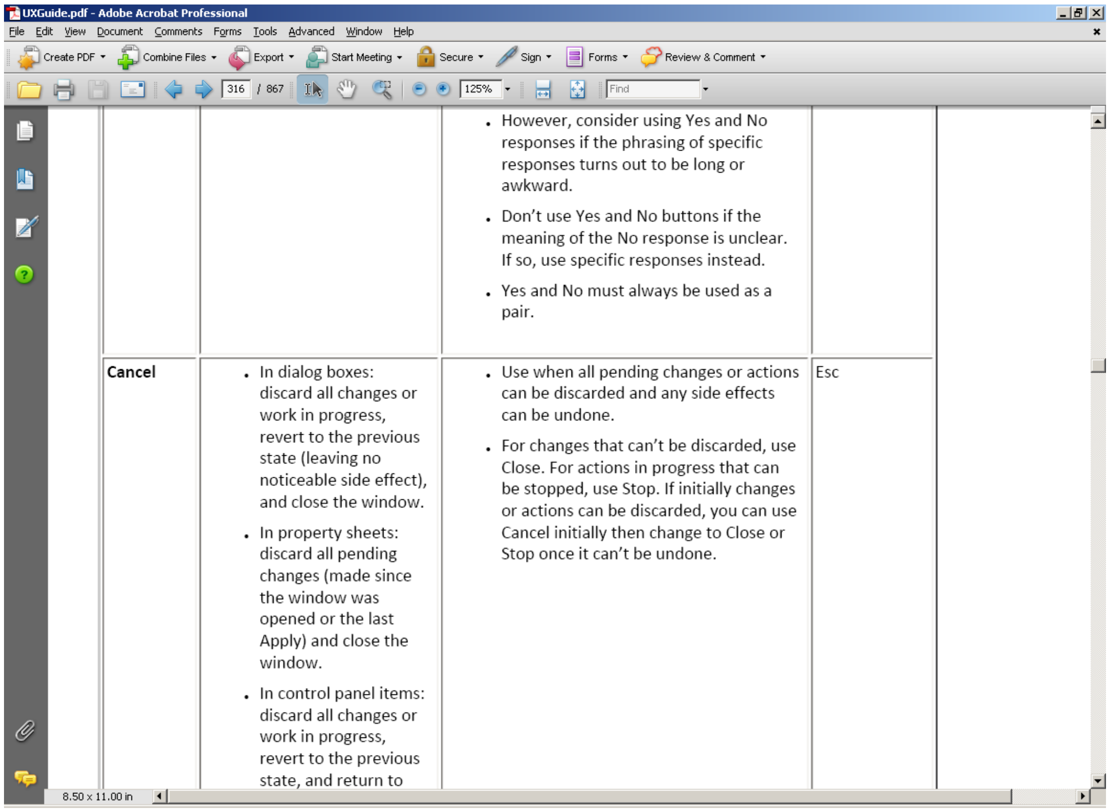
© 2010, Microsoft Corporation. All rights reserved. Page 315 of 86

Figure 3-28: Supplemental explanations


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools Advanced Window Help
Create PDF Combine Files Export Start Meeting Secure Sign Forms Review & Comment
316 / 867 125% Find
8.50 x 11.00 in

OK	• In dialog boxes: apply the changes or commit to the task and close the window. • In owner property windows: apply the pending changes (made since the window was opened or the last Apply) and close the window. • In owned property windows: keep the changes, close the window, and apply the changes when the owner window's changes are applied.	• Use with windows that aren't task specific, such as property sheets. • For windows used to perform one specific task, use a specific label instead that starts with a verb (example: Print). • For windows in which users can't make changes, use Close.	Enter
Yes/No	Yes is the affirmative response to a yes or no question, whereas No is the negative response.	• Use Yes and No buttons only to respond to yes or no questions. Never use OK and Cancel for yes or no questions. • Prefer specific responses over Yes and No buttons. While there's nothing wrong with using Yes and No, specific responses can be understood more quickly, resulting in efficient decision making.	'Y' and 'N'

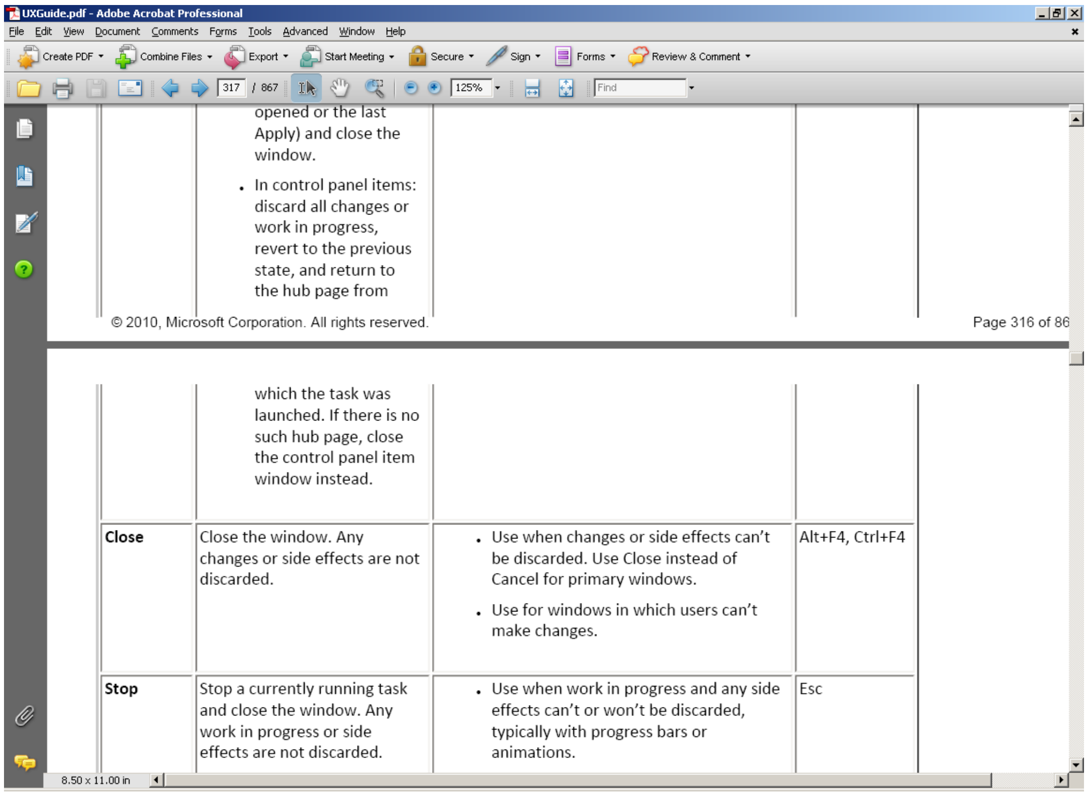
Figure 3-29: Supplemental explanations



UXGuide.pdf - Adobe Acrobat Professional

		• However, consider using Yes and No responses if the phrasing of specific responses turns out to be long or awkward. • Don't use Yes and No buttons if the meaning of the No response is unclear. If so, use specific responses instead. • Yes and No must always be used as a pair.	
Cancel	• In dialog boxes: discard all changes or work in progress, revert to the previous state (leaving no noticeable side effect), and close the window. • In property sheets: discard all pending changes (made since the window was opened or the last Apply) and close the window. • In control panel items: discard all changes or work in progress, revert to the previous state, and return to	• Use when all pending changes or actions can be discarded and any side effects can be undone. • For changes that can't be discarded, use Close. For actions in progress that can be stopped, use Stop. If initially changes or actions can be discarded, you can use Cancel initially then change to Close or Stop once it can't be undone.	Esc

Figure 3-30: Supplemental explanations



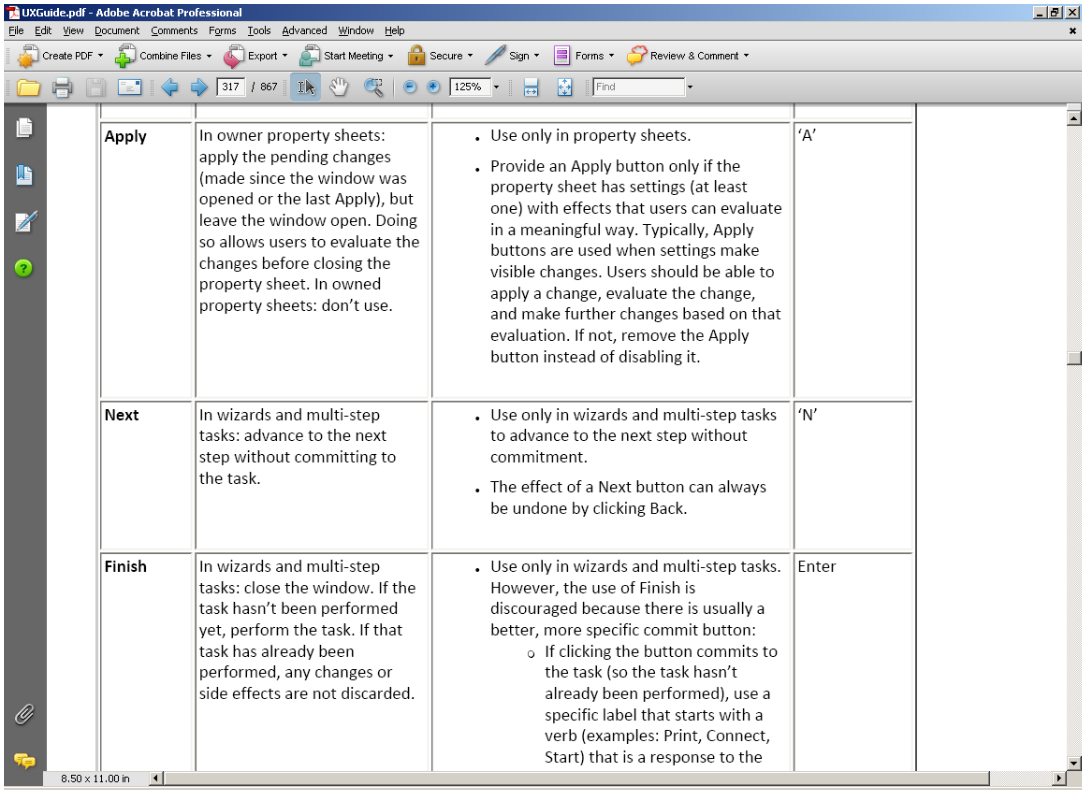
UXGuide.pdf - Adobe Acrobat Professional

	opened or the last Apply) and close the window. • In control panel items: discard all changes or work in progress, revert to the previous state, and return to the hub page from		

© 2010, Microsoft Corporation. All rights reserved. Page 316 of 86

	which the task was launched. If there is no such hub page, close the control panel item window instead.		
Close	Close the window. Any changes or side effects are not discarded.	• Use when changes or side effects can't be discarded. Use Close instead of Cancel for primary windows. • Use for windows in which users can't make changes.	Alt+F4, Ctrl+F4
Stop	Stop a currently running task and close the window. Any work in progress or side effects are not discarded.	• Use when work in progress and any side effects can't or won't be discarded, typically with progress bars or animations.	Esc

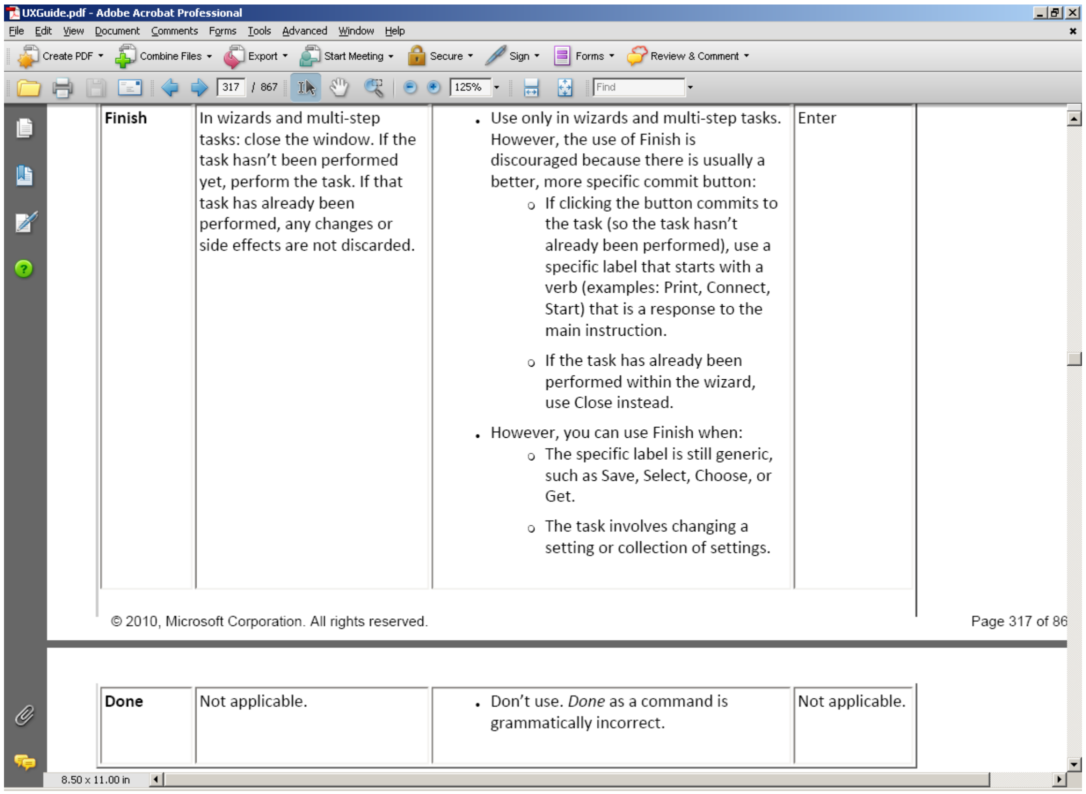
Figure 3-31: Supplemental explanations



UXGuide.pdf - Adobe Acrobat Professional

Apply	In owner property sheets: apply the pending changes (made since the window was opened or the last Apply), but leave the window open. Doing so allows users to evaluate the changes before closing the property sheet. In owned property sheets: don't use.	• Use only in property sheets. • Provide an Apply button only if the property sheet has settings (at least one) with effects that users can evaluate in a meaningful way. Typically, Apply buttons are used when settings make visible changes. Users should be able to apply a change, evaluate the change, and make further changes based on that evaluation. If not, remove the Apply button instead of disabling it.	'A'
Next	In wizards and multi-step tasks: advance to the next step without committing to the task.	• Use only in wizards and multi-step tasks to advance to the next step without commitment. • The effect of a Next button can always be undone by clicking Back.	'N'
Finish	In wizards and multi-step tasks: close the window. If the task hasn't been performed yet, perform the task. If that task has already been performed, any changes or side effects are not discarded.	• Use only in wizards and multi-step tasks. However, the use of Finish is discouraged because there is usually a better, more specific commit button: o If clicking the button commits to the task (so the task hasn't already been performed), use a specific label that starts with a verb (examples: Print, Connect, Start) that is a response to the	Enter

Figure 3-32: Supplemental explanations



UXGuide.pdf - Adobe Acrobat Professional

Finish	In wizards and multi-step tasks: close the window. If the task hasn't been performed yet, perform the task. If that task has already been performed, any changes or side effects are not discarded.	• Use only in wizards and multi-step tasks. However, the use of Finish is discouraged because there is usually a better, more specific commit button: o If clicking the button commits to the task (so the task hasn't already been performed), use a specific label that starts with a verb (examples: Print, Connect, Start) that is a response to the main instruction. o If the task has already been performed within the wizard, use Close instead. • However, you can use Finish when: o The specific label is still generic, such as Save, Select, Choose, or Get. o The task involves changing a setting or collection of settings.	Enter

© 2010, Microsoft Corporation. All rights reserved. Page 317 of 86

Done	Not applicable.	• Don't use. *Done* as a command is grammatically incorrect.	Not applicable.

Figure 3-33: Supplemental explanations

3.33 Style and Tone (Page 319)

3.33.1 Guidelines

3.33.1.1 Use The Windows Tone

Tone in your program should be:

- **Accurate.** Users should feel reassured that the information is technically accurate. If the information isn't accurate, the user's experience with that specific task is spoiled, and he loses faith in any other assistance he reads from that source.
- **Encouraging.** Use language that conveys that the software empowers users to do things, rather than allows them to do things. For example, use "you can" rather than "Windows lets you" or "this feature allows you." (Exception: it's okay to use "allow" when referring to features—such as security features—that permit or deny an action.)
- **Insightful.** Users should believe that you (and by extension your application) know when a certain task is complicated and that you will guide them through it. At the same time, treat users as intelligent people who happen to need help with a particular problem.
- **Objective.** Sometimes users want a richer explanation; often though they just want to know what they need to move on. This requires objectivity—to recognize that the goal (productivity, curiosity, enjoyment) is the user's goal, not the writer's. It also requires that you shed any predisposed notions about the user.
- **User-focused.** Write from the user's perspective and preferably from the perspective of what you can do for the user. Users should feel that they will find information that is relevant and accessible to them.
  - **Light.** A light voice feels easy to deal with and quick to comprehend. It isn't difficult or burdensome, and it avoids straining to be profound or serious.
  - **Inspiring.** An inspiring voice motivates users, stimulating them to take action. A certain enthusiasm marks this kind of voice, giving it a more engaging personality than the machine-like tone users have often come to expect from their computers.
  - **Straightforward.** A straightforward voice is candid and open, free from pretense or deceit.
  - **Trustworthy.** A trustworthy voice is worthy of confidence, reliable, accurate, and honest. It only takes only a few inaccuracies to lose the user's trust, but it may take a long time to earn that trust back.

By contrast, be sure to avoid the following tones, which are much more likely to receive a negative reaction:

- **Machine tone.** Feels like having an impersonal, inflexible exchange with a computer or robot.
- **Corporate tone.** Feels like having a monologue with an all-powerful, all-knowing, impersonal corporation.
- **Law enforcement tone.** Feels like being interrogated with a barrage of intrusive questions.
- **Lawyerly tone.** Feels like being asked to perform some legally significant act.
- **Sales rep tone.** Feels like being asked to buy or try something, with any concerns being glossed over.
- **Superior, condescending, or angry tone.** Feels like the software is belittling users, talking down to them, or upset with them. Typically, this tone is very technical, draws unnecessary attention to the user's mistakes, and feels rude.
- **Boastful tone.** Feels like the software is bragging about its accomplishments or otherwise drawing too much attention to itself.
- **Flippant tone.** Feels like users' goals and emotions aren't being taken seriously, or their use of the program is being taken for granted.

3.33.1.2 Use Real-World Language

- **Use everyday words when you can and avoid words you wouldn't say to someone else in person.** 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.
  - **Use short, plain words whenever possible.** Shorter words are more conversational, save space on screen, and are easier to scan.
  - **Don't invent words or apply new meanings to standard words.** Assume that users are more familiar with a word's established meaning than with a special meaning given it by the technology industry. When an industry term is required, provide an in-context definition. Avoid jargon, but remember that some expressions specific to computer usage—hacker, burn a CD, and so on—are already part of everyday speech.
  - **Don't use symbols as a substitute for simple words.**

3.33.1.3 Be Precise

Choose words with a clear meaning.

- Omit needless words—don't use two or three words when one will do.
- Avoid unnecessary adverbs.
- Choose single-word verbs over multi-word verbs.
- Don't convert verbs to nouns and nouns to verbs.

3.33.1.4 Be Consistent

- Consistent terminology promotes learning and a better understanding of technical concepts. Inconsistency forces users to figure out whether different words and actions mean the same thing.
- Consistent syntax helps set users' expectations. Once these expectations are set, users can more quickly parse text that uses consistent syntax. For example, if instructions are always written in the imperative form, users will learn to pay closer attention to imperative sentences.

3.33.1.5 Contractions

Contractions lend a shorter, snappier, more conversational rhythm to writing. Use them as appropriate and in context. Don’t use contractions with product names or other proper nouns.

3.33.1.6 Colloquialisms, Idioms, And Slang

Consider using colloquialisms or slang only in special situations, such as product tours, setup screens, or content that won't be localized. Recent studies have shown that users enjoy the unexpected word or familiar phrase. Bear in mind that using colloquialisms and slang can be difficult and costly to translate effectively, so such language is best used judiciously.

3.33.1.7 Person

- Address the user as *you*, directly or indirectly.
- Use the second person (you, your) to tell users what to do. Often the second person is implied.
  - Use the first person (I, me, my) to let users tell the program what to do.
  - Use “we” judiciously. The first-person plural can suggest a daunting corporate presence. However, it can be preferable to using the name of your application. Use “we recommend” rather than “it is recommended.”
  - Avoid third-person references (the user) because they create a more formal, less personal tone.

3.33.1.8 Voice

- **Use the active voice,** which emphasizes the person or thing doing the action. It is more direct and personal than the passive voice, which can be confusing or sound formal.
- Use the passive voice only to avoid a wordy or awkward construction; when the action rather than the doer is the focus of the sentence; when the subject is unknown; or in error messages, when the user is the subject and might feel blamed for the error if the active voice were used.

- **Phrase statements in the positive form**, and emphasize what users can accomplish, rather than what they can't.

3.33.1.9 Attitude Toward The User

- **Be polite, supportive, and encouraging.** The user should never feel condescended to, blamed, or intimidated.
  - **Strike the right balance: be warm toward the user without being too intimate or too business-like.** Imagine that you are helping a friend use the product for the first time. This person is not your best friend or significant other, but instead, a neighbor or family friend. Users should feel comfortable and at home when using your program, but the language should not feel presumptuous or too familiar.
    - **Limit *please* to situations that inconvenience the user in some way,** such as: The user is asked to do something inconvenient, such as waiting, repeating a task or updating a program.
    - The user can't complete a task because of a missing feature, design flaw, or program bug.
    - The user has gone out of his or her way to be helpful, as by participating in a customer feedback program or filing a bug report.
  - You should also use please whenever its absence would be considered curt.
- **Use *sorry* only in error messages that result in serious problems for the user** (for example, data loss, the user can't continue to use the computer, or the user must get help from a technical representative). Don't apologize if the issue occurred during the normal functioning of the program (for example, if the user needs to wait for a network connection to be found).

3.33.1.10 Sentence Structure And Length

- Because users often scan text, **make every word count.** Simple, concise sentences (and paragraphs) not only save space on the screen but are the most effective means of conveying that an idea or action is important. Use your best judgment—make sentences tight, but not so tight that the tone seems abrupt and unfriendly.
- **Avoid repetition.** Review each window and eliminate duplicate words and statements. Don't avoid important text—be explicit whenever necessary—but don't be redundant and don't explain things that go without saying.
  - **Use sentence fragments if appropriate.** Sentence fragments are short and punchy—and, as they typically take the interrogative form, they are a good way of directly engaging the user.
  - **Start sentences with conjunctions** *(and, but, or)* if you need to.
  - **Substitute lists and tables for complex sentences.** Lists (whether numbered or bulleted) and tables are clearer and easier to scan.

  - **Use parallel grammatical constructions.** Parallelism requires that words and phrases that have the same function have the same form. Use parallel language whenever you express ideas of equal weight, and for UI elements that are parallel in function (such as headings, labels, lists, or page titles).

3.34 Messages (Page 326)

3.34.1 Guidelines

3.34.1.1 Presentation

- **Use task dialogs whenever appropriate** to achieve a consistent look and layout. Task dialogs require Windows Vista® or later, so they aren't suitable for earlier versions of Windows. If you must use a message box, separate the main instruction from the supplemental instruction with two line breaks.

3.34.1.2 User Input Errors

- Whenever possible, prevent or reduce user input errors by: Using controls that are constrained to valid values.
- Disabling controls and menu items when clicking would result in error, as long as it's obvious why the control or menu item is disabled.
- Providing good default values.
  - Use modeless error handling (in-place errors or balloons) for contextual user input problems.
  - Use balloons for non-critical, single-point user input problems detected while in a text box or immediately after a text box loses focus. Balloons don't require available screen space or the dynamic layout that is required to display in-place messages. Display only a single balloon at a time. Because the problem isn't critical, no error icon is necessary. Balloons go away when clicked, when the problem is resolved, or after a timeout. *In this*
  - Use in-place errors for delayed error detection, usually errors found by clicking a commit button. (Don't use in-place errors for settings that are immediately committed.) There can be multiple in-place errors at a time. Use normal text and a 16 × 16 pixel error icon, placing them directly next to the problem whenever possible. In-place errors don't go away unless the user commits and no other errors are found.
  - Use modal error handling (task dialogs or message boxes) for all other problems, including errors that involve multiple controls or are noncontextual or noninput errors found by clicking a commit button.

  - When a user input problem is reported, set input focus to the first control with the incorrect data. Scroll the control into view if necessary. If the control is a text box, select the entire contents. It should always be obvious what the error message is referring to.
  - Don't clear incorrect input. Instead, leave it so that the user can see and correct the problem without starting over. Exception: Clear incorrect password and PIN text boxes because users can't correct masked input effectively.

3.34.1.3 Troubleshooting

- **Avoid creating troubleshooting problems.** Don't rely on a single error message to report a problem with several different detectable causes.
- **Use a different error message (typically a different supplemental instruction) for each detectable cause.** For example, if a file cannot be opened for several reasons, provide a separate supplemental instruction for each reason.
- **Use a message with multiple causes only when the specific cause can't be determined.** In this case, present the solutions in order of likelihood of fixing the problem. Doing so helps users solve the problem more efficiently.

3.34.1.4 Icons

- **Modal error message dialogs don't have title bar icons.** Title bar icons are used as a visual distinction between primary windows and secondary windows.

  Use an error icon. Exceptions: If the error is a user input problem displayed using a modal dialog box or balloon, don't use an icon. Doing so is counter to the encouraging tone of Windows. However, in-place error messages should use a small error icon (16 × 16 pixel) to clearly identify them as error messages.

  - If the problem is for a feature that has an icon (and not a user input problem), you can use the feature icon with an error overlay. If you do this, also use the feature name as the error's subject.
    - **Don't use warning icons for errors.** This is often done to make the presentation feel less severe. Errors aren't warnings.

3.34.1.5 Progressive Disclosure

- **Use a Show/Hide details progressive disclosure button to hide advanced or detailed information in an error message.** Doing so simplifies the error message for typical usage. Don't hide needed information, because users might not find it.
  - **Don't use Show/Hide details unless there really is more detail.** Don't just restate the existing information in a more verbose format.
  - **Don't use Show/Hide details to show Help information.** Use Help links instead.

3.34.1.6 Don’t Show This Message Again

If an error message needs this option, reconsider the error and its frequency. If it has all the characteristics of a good error (relevant, actionable, and infrequent), it shouldn’t make sense for users to suppress it.

3.34.2 Default Values

Select the safest, least destructive, or most secure response to be the default. If safety isn’t a factor, select the most likely or convenient command.

3.34.3 Help

- **Design error messages to avoid the need for Help.** Ordinarily users shouldn’t have to read external text to understand and solve the problem, unless the solution requires several steps.
- **Make sure the Help content is relevant and helpful.** It shouldn’t be a verbose restatement of the error message—rather, it should contain useful information that is beyond the scope of the error message, such as ways to avoid the problem in the future. Don’t provide a Help link just because you can.
- **Use specific, concise, relevant Help links to access Help content.** Don’t use command buttons or progressive disclosure for this purpose.
- **For error messages that you can’t make specific and actionable, consider providing links to online Help content.** By doing so, you can provide users with additional information that you can update after the program is released.

3.34.4 Error Codes

- For error messages that you can’t make specific and actionable or they benefit from Help, consider also providing error codes. Users often use these error codes to search the Internet for additional information.
- Always provide a text description of the problem and solution. Don’t depend just on the error code for this purpose.
  - Assign a unique error code for each different cause. Doing so avoids troubleshooting.
  - Choose error codes that are easily searchable on the Internet. If you use 32-bit codes, use a hexadecimal representation with a leading “0x” and uppercase characters.
  - Use Show/Hide details to display error codes. Phrase as *Error code: <error code>*.

3.34.5 Sound

Don’t accompany error messages with a sound effect or beep. Doing so is jarring and unnecessary. **Exception:** Play the Critical Stop sound effect if the problem is critical to the operation of the computer, and the user must take immediate action to prevent serious consequences.

3.34.6 Text

3.34.6.1 General

- **Remove redundant text.** Look for it in titles, main instructions, supplemental instructions, command links, and commit buttons. Generally, leave full text in instructions and interactive controls, and remove any redundancy from the other places.
- **Use user-centered explanations.** Describe the problem in terms of user actions or goals, not in terms of what the software is unhappy with. Use language that the target users understand and use. Avoid technical jargon.
- Don’t use the following words:
  - Error, failure (use “problem” instead)
  - Failed to (use “unable to” instead)
  - Illegal, invalid, bad (use “incorrect” instead)
  - Abort, kill, terminate (use “stop” instead)
  - Catastrophic, fatal (use “serious” instead)
- 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.
- **Be specific.** Avoid vague wording, such as *syntax error* and *illegal operation*. Provide specific names, locations, and values of the objects involved.
- **Don’t give possibly unlikely problems, causes, or solutions in an attempt to be specific.** Don’t provide a problem, cause, or solution unless it is likely to be right. For example, it is better to say “An unknown error occurred” than something that is likely to be inaccurate.
- **Avoid the word “please,”** except in situations in which the user is asked to do something inconvenient (such as waiting) or the software is to blame for the situation.

  - **Use the word "sorry" only in error messages that result in serious problems for the user** (for example, data loss or inability to use the computer). Don't apologize if the issue occurred during the normal functioning of the program (for example, if the user needs to wait for a network connection to be found). **Correct:** We're sorry, but Fabrikam Backup detected an unrecoverable problem and was shut down to protect files on your computer.
  - **Refer to products using their short names.** Don't use full product names or trademark symbols. Don't include the company name unless users associate the company name with the product. Don't include program version numbers.
  - **Use double quotation marks around object names.** Doing so makes the text easier to parse and avoids potentially embarrassing statements. **Exception:** Fully qualified file paths, URLs, and domain names don't need to be in double quotation marks.
  - **Avoid starting sentences with object names.** Doing so is often difficult to parse.
  - **Don't use exclamation marks or words with all capital letters.** Exclamation marks and capital letters make it feel like you are shouting at the user.

3.34.6.2 Titles

- **Use the title to identify the command or feature from which the error originated.** Exceptions: If an error is displayed by many different commands, consider using the program name instead.
- If that title would be redundant or confusing with the main instruction, use the program name instead.
- **Don't use the title to explain or summarize the problem**—that's the purpose of the main instruction.
  - Use **title-style capitalization**, without ending punctuation.

3.34.6.3 Main Instructions

- Use the main instruction to describe the problem in clear, plain, specific language.
- Be concise—use only a single, complete sentence. Pare the main instruction down to the essential information. You can leave the subject implicit if it is your program or the user. Include the reason for the problem if you can do so concisely. If you must explain anything more, use a supplemental instruction.
- Be specific—if there are objects involved, give their names.

- Avoid putting full file paths and URLs in the main instruction. Rather, use a short name (such as the file name) and put the full name (such as the file path) in the supplemental instruction. However, you can put a single full file path or URL in the main instruction if the error message doesn’t otherwise need a supplemental instruction.
- Don’t give the full file path and URL at all if it’s obvious from the context.
- Use present tense whenever possible.
- Use sentence-style capitalization.
- Don’t include final periods if the instruction is a statement. If the instruction is a question, include a final question mark.

3.34.6.4 Main Instruction Templates

While there are no strict rules for phrasing, try using the following main instruction templates whenever possible:

- <optional subject name> can’t <perform action>
- <optional subject name> can’t <perform action> because <reason>
- <optional subject name> can’t <perform action> to “<object name>”
- <optional subject name> can’t <perform action> to “<object name>” because <reason>
- There is not enough <resource> to <perform action>
- <Subject name> doesn’t have a <object name> required for <purpose>
- <Device or setting> is turned off so that <undesired results>
- <Device or setting> isn’t <available | found | turned on | enabled>
- “<object name>” is currently unavailable
- The user name or password is incorrect
- You don’t have permission to access “<object name>”
- You don’t have privilege to <perform action>
- <program name> has experienced a serious problem and must close immediately

Of course, make changes as needed for the main instruction to be grammatically correct and comply with the main instruction guidelines.

3.34.6.5 Supplemental Instructions

- Use the supplemental instruction to: Give additional details about the problem.
- Explain the cause of the problem.

- List steps the user can take to fix the problem.
- Provide measures to prevent the problem from reoccurring.

Whenever possible, propose a practical, helpful 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.

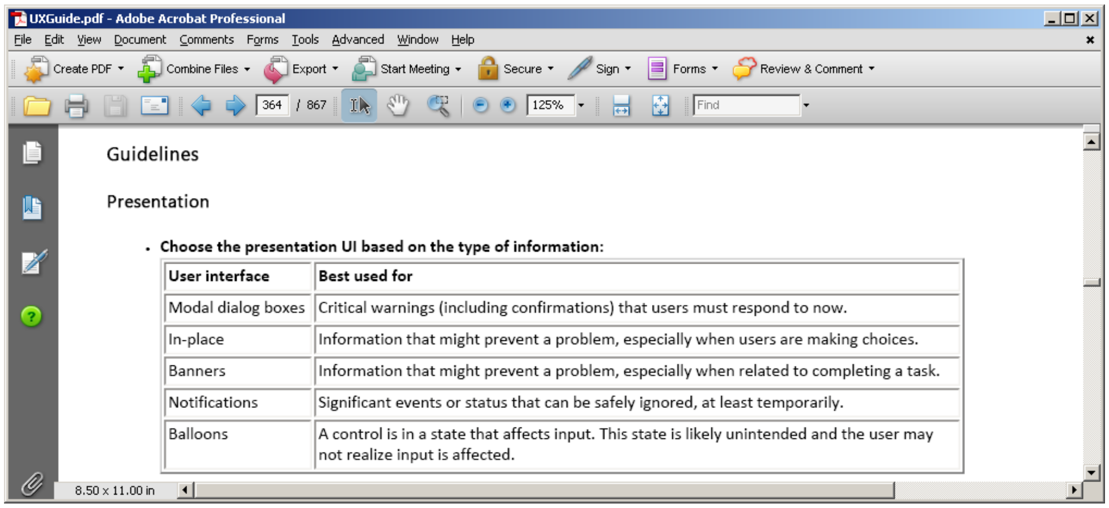
- If the problem is an incorrect value that the user entered, use the supplemental instruction to explain the correct values. Users shouldn't have to determine this information from another source.
- Don't provide a solution if it can be trivially deduced from the problem statement.
- If the solution has multiple steps, present the steps in the order in which they should be completed. However, avoid multi-step solutions because users have difficulty remembering more than two or three simple steps. If more steps are required, refer to the appropriate Help topic.
- Keep supplemental instructions concise. Provide only what users need to know. Omit unnecessary details. Aim for a maximum of three sentences of moderate length.
- To avoid mistakes while users perform instructions, put the results before the action.
- Don't recommend contacting an administrator unless doing so is among the most likely solutions to the problem. Reserve such solutions for problems that really can only be solved by an administrator.
- Don't recommend contacting technical support. The option to contact technical support to solve a problem is always available, and doesn't need to be promoted through error messages. Instead, focus on writing helpful error messages so that users can solve problems without contacting technical support.
- Use complete sentences, sentence-style capitalization, and ending punctuation.

3.34.6.6 Commit Buttons

- If the error message provides command buttons or command links that solve the problem, follow their respective guidelines in **Dialog Boxes**.
- If the program must terminate as a result of the error, provide an Exit program button. To avoid confusion, don't use Close for this purpose.

Otherwise, provide a Close button. Don't use OK for error messages, because this wording implies that problems are OK. **Exception:** Use OK if your error reporting mechanism has fixed labels (as with the MessageBox API.)

3.35 Warning Messages (Page 357)


UXGuide.pdf - Adobe Acrobat Professional
File Edit View Document Comments Forms Tools Advanced Window Help
Create PDF
Combine Files
Export
Start Meeting
Secure
Sign
Forms
Review & Comment
364 / 867
125%
Find
8.50 x 11.00 in

Guidelines

Presentation

- **Choose the presentation UI based on the type of information:**

User interface	**Best used for**
Modal dialog boxes	Critical warnings (including confirmations) that users must respond to now.
In-place	Information that might prevent a problem, especially when users are making choices.
Banners	Information that might prevent a problem, especially when related to completing a task.
Notifications	Significant events or status that can be safely ignored, at least temporarily.
Balloons	A control is in a state that affects input. This state is likely unintended and the user may not realize input is affected.

Figure 3-34: Guidelines

For modal dialog boxes:

- Use task dialogs whenever appropriate to achieve a consistent look and layout. Task dialogs require Windows Vista® or later, so they aren't suitable for earlier versions of Windows.
- Display only one warning message per condition. For example, display a single warning that completely explains a condition instead of describing it one detail at a time per message. Displaying a sequence of warning dialogs for a single condition is confusing and annoying.
- Don't display a warning more than once per condition. Constant warnings quickly become ineffective and annoying. Users often become more focused on getting rid of the warning than addressing the problem. If you must warn repeatedly for a single condition, use progressive escalation.
- Don't accompany warnings with a sound effect or beep. Doing so is jarring and unnecessary. Exception: If the user must respond immediately, you can use a sound effect.

3.35.1 Icons

- Don't place a warning icon in the title bar of a dialog box.
- **Use a warning icon.** Exceptions: If the warning is for a feature that has an icon, you can use the feature icon with a warning overlay.

3.35.2 Don’t Show This Message Again

If a warning dialog box needs this option, reconsider the warning and its frequency. If it has all the characteristics of a good warning (involves risk, and is immediately relevant, actionable, not obvious, and infrequent), it shouldn’t make sense for users to suppress it.

3.35.3 Progressive Disclosure

- **If you must include advanced information in a warning message, reveal it by using progressive disclosure buttons** (for example, “Show details”). Doing so simplifies the warning for typical usage. Don’t hide needed information because users might not find it.
- **Don’t use “Show details” unless there really is more detail.** Don’t just restate the existing information in a different format.

3.35.4 Default Values

Select the safest, least destructive, or most secure response to be the default.

3.35.5 Text

3.35.5.1 General

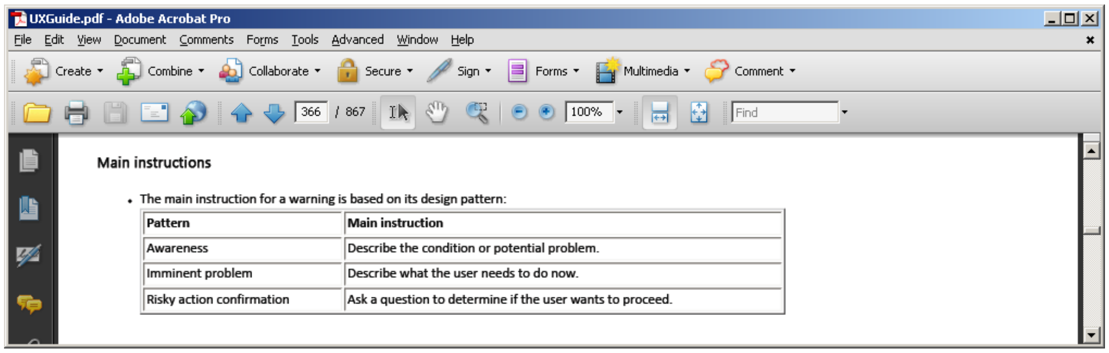
- **Remove redundant text.** Look for it in titles, main instructions, supplemental instructions, content areas, command links, and commit buttons. Generally, leave full text in instructions and interactive controls, and remove any redundancy from the other places.
- **Don’t use the terms “warning” or “caution” in the text.** When **used correctly**, the warning icon sufficiently communicates that users must proceed with caution.

3.35.5.2 Titles

Use the title to identify the command or feature where the warning came from. Exceptions: If a warning is displayed by many different commands, consider using the program name instead.

If that title would be redundant or confusing with the main instruction, use the program name instead.

- Don’t use the title to explain what to do in the dialog—that’s the purpose of the main instruction.
- Use title-style capitalization, without ending punctuation.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
366 / 867
100%
Find

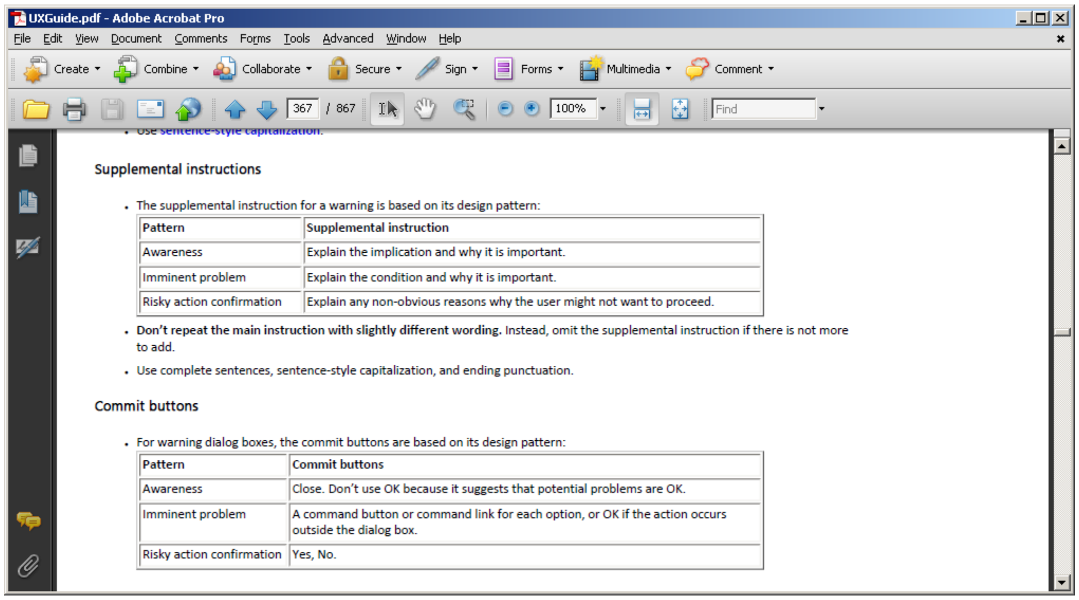
Main instructions

- The main instruction for a warning is based on its design pattern:

Pattern	Main instruction
Awareness	Describe the condition or potential problem.
Imminent problem	Describe what the user needs to do now.
Risky action confirmation	Ask a question to determine if the user wants to proceed.

Figure 3-35: Main instructions

- **Be concise—use only a single, complete sentence.** Strip the main instruction down to the essential information. If you must explain anything more, use a supplemental instruction.
- **Use words like “now” and “immediately” if the user must act immediately.** Don’t use these words if there is no urgency.
- **Be specific—if there are objects involved, give their full names.**
- Use **sentence-style capitalization**.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
367 / 867
100%
Find

- Use sentence-style capitalization.

Supplemental instructions

- The supplemental instruction for a warning is based on its design pattern:

Pattern	Supplemental instruction
Awareness	Explain the implication and why it is important.
Imminent problem	Explain the condition and why it is important.
Risky action confirmation	Explain any non-obvious reasons why the user might not want to proceed.

- **Don't repeat the main instruction with slightly different wording.** Instead, omit the supplemental instruction if there is not more to add.
- Use complete sentences, sentence-style capitalization, and ending punctuation.

Commit buttons

- For warning dialog boxes, the commit buttons are based on its design pattern:

Pattern	Commit buttons
Awareness	Close. Don't use OK because it suggests that potential problems are OK.
Imminent problem	A command button or command link for each option, or OK if the action occurs outside the dialog box.
Risky action confirmation	Yes, No.

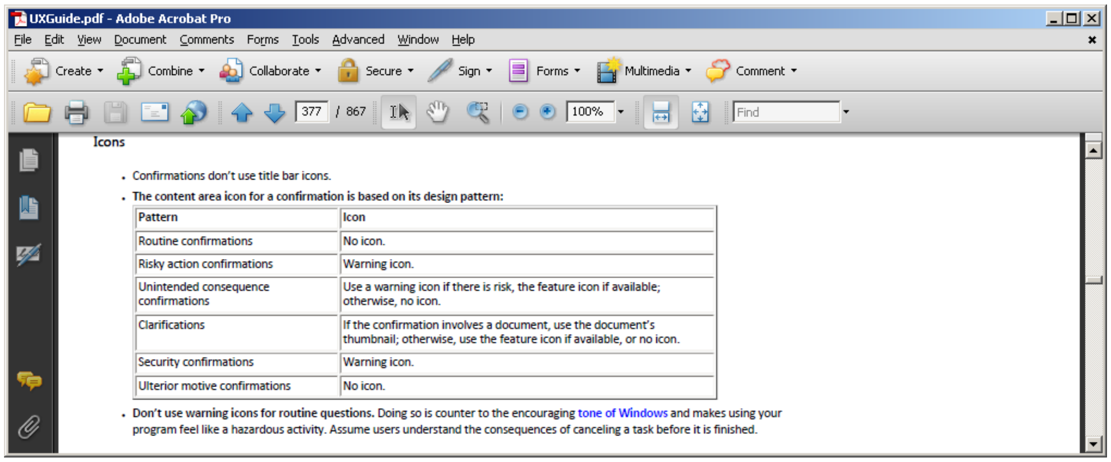
Figure 3-36: Supplemental instructions

3.36 Confirmations (Page 369)

3.36.1 Guidelines

3.36.1.1 General

Use "Save changes" confirmations only when there are significant changes. Don't confirm changes that weren't directly made by the user, such as automatic document reformatting.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
377 / 867 100% Find

Icons

- Confirmations don't use title bar icons.
- The content area icon for a confirmation is based on its design pattern:

Pattern	Icon
Routine confirmations	No icon.
Risky action confirmations	Warning icon.
Unintended consequence confirmations	Use a warning icon if there is risk, the feature icon if available; otherwise, no icon.
Clarifications	If the confirmation involves a document, use the document's thumbnail; otherwise, use the feature icon if available, or no icon.
Security confirmations	Warning icon.
Ulterior motive confirmations	No icon.

- Don't use warning icons for routine questions. Doing so is counter to the encouraging tone of Windows and makes using your program feel like a hazardous activity. Assume users understand the consequences of canceling a task before it is finished.

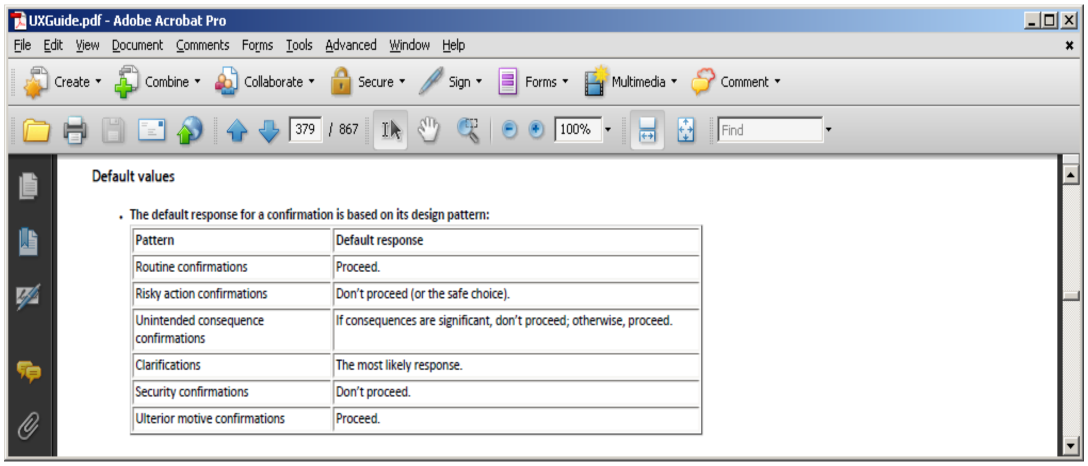
Figure 3-37: Icons

3.36.1.2 Commit Buttons

- Use specific responses to the main instruction if the reason for the confirmation is obvious or can be made self explanatory. Otherwise, use Yes and No buttons for confirmation responses. Doing so makes users give the confirmation some thought before responding.
- Never use OK and Cancel for confirmations.
- To close a program or restart Windows, use specific responses to the main instruction. To prevent any misunderstanding, don't use Close or Yes/No for this purpose.

3.36.1.3 Command Links

- For the clarifications pattern, consider using command links to make the alternatives clear.
- Present the most commonly used command links first. The resulting order should roughly follow the likelihood of use, but also have a logical flow.
- If a command link requires further explanation, provide a supplemental explanation. Supplemental explanations describe why users might want to choose the option or what happens if the option is chosen.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
379 / 867
100%
Find

Default values

- The default response for a confirmation is based on its design pattern:

Pattern	Default response
Routine confirmations	Proceed.
Risky action confirmations	Don't proceed (or the safe choice).
Unintended consequence confirmations	If consequences are significant, don't proceed; otherwise, proceed.
Clarifications	The most likely response.
Security confirmations	Don't proceed.
Ulterior motive confirmations	Proceed.

Figure 3-38: Default values

3.36.1.4 Don’t Show This Message Again

- **Use this option only for the routine and ulterior motive confirmation patterns.** For the other patterns, if the information is necessary, it should always be displayed.
- **Don’t provide this option to justify displaying an unnecessary confirmation.** Just get rid of the confirmation instead.

3.36.1.5 Bulk Operations

- For confirmations that apply to bulk operations, provide an option to apply the confirmation to the entire operation.
- Eliminate or postpone confirmations in a bulk operation.

3.36.1.6 Progressive Disclosure

- If you must include advanced information in a confirmation message, reveal it by using progressive disclosure buttons (for example, “Show details”). Doing so simplifies the confirmation for typical usage. Don’t hide needed information because users might not find it.
- Don’t use “Show details” unless there really is more detail. Don’t just restate the existing information in a different format.

3.36.1.7 User Account Control

- **Don’t use the User Account Control (UAC) elevation UI as a substitute for a confirmation.** If an action needs a confirmation, use a separate dialog box. During the elevation UI, users need to focus on whether they started task and if the program is trustworthy.

- **Display the confirmation before the elevation UI.** Doing so eliminates unnecessary elevations.

3.36.2 Text

3.36.2.1 General

- **Remove redundant text.** Look for redundant text in titles, main instructions, supplemental instructions, content areas, command links, and commit buttons. Generally, leave full text in instructions and interactive controls, and remove any redundancy from the other places.
- **Don't use "warning" or "caution" in the text.** If users need to proceed with caution, indicate this using a **warning icon** instead.

3.36.2.2 Titles

Use the title to identify the command or feature where the confirmation came from. Exceptions: If a confirmation is displayed by many different commands, consider using the program name instead.

- If that title would be redundant or confusing with the main instruction, use the program name instead.
- However, if the confirmation is from a long-running task and may display well after the task started, always use the command or feature to clearly identify the context.
- **Don't use the title to explain what to do in the dialog—**that's the purpose of the main instruction.
- If it adds clarity, start the title with the word "Confirm."

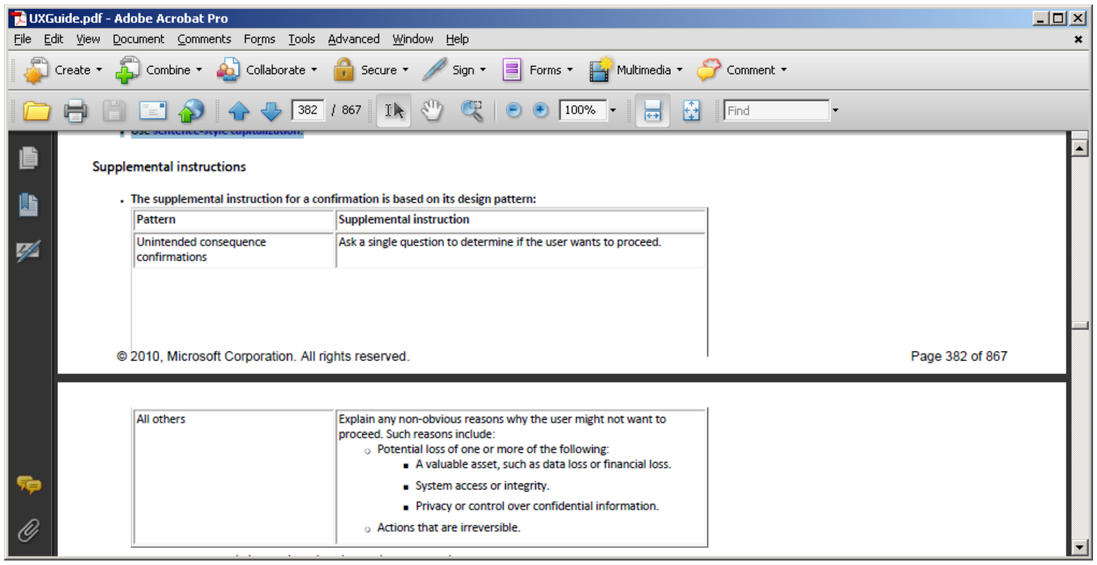
For risky action confirmations, you may add the name of the object involved for extra emphasis.

- Use **title-style capitalization**, without ending punctuation.

3.36.2.3 Main Instructions

- **Be concise—use only a single, complete sentence.** Strip the main instruction down to the essential information. If you must explain anything more, use a supplemental instruction.
- **Be specific—if there are objects involved, give their full names.**
- **Use positive phrasing.** Positive phrasing is easier for users to understand.
- For risky action confirmations, use the term "permanently" to indicate that an action can't be undone.

- Use **sentence-style capitalization**.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
382 / 867
100%
Find

Supplemental instructions

- The supplemental instruction for a confirmation is based on its design pattern:

Pattern	Supplemental instruction
Unintended consequence confirmations	Ask a single question to determine if the user wants to proceed.
All others	Explain any non-obvious reasons why the user might not want to proceed. Such reasons include: ○ Potential loss of one or more of the following: ▪ A valuable asset, such as data loss or financial loss. ▪ System access or integrity. ▪ Privacy or control over confidential information. ○ Actions that are irreversible.

© 2010, Microsoft Corporation. All rights reserved. Page 382 of 867

Figure 3-39: Supplemental instructions

- **Don’t repeat the main instruction with slightly different wording.** Instead, omit the supplemental instruction if there is not more to add.
- **For unintended consequence confirmations, consider using the term “**anyway” **to concisely indicate that there is a reason not to continue** in case the user overlooked the main instruction. See **Design concepts** for more information.
- Use complete sentences, sentence-style capitalization, and ending punctuation.

3.37 Notifications (Page 384)

3.37.1 Guidelines

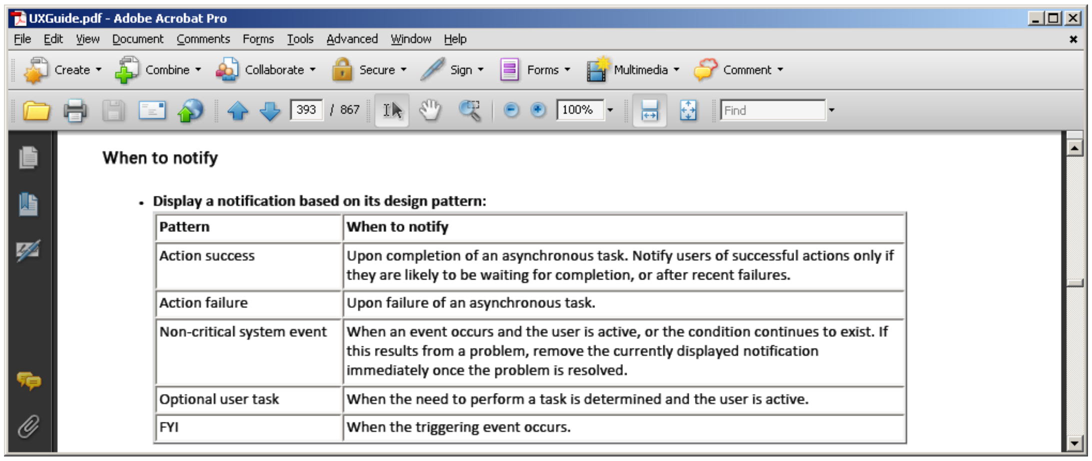
3.37.1.1 General

- **Select the notification pattern based on its usage.** For a description of each usage pattern, see the previous table.
- **Don’t use any notifications during the initial Windows experience.** To improve its first experience, Windows 7 suppresses all notifications displayed during the first few hours of usage. Design your program assuming users won’t see any such notifications.

3.37.1.2 What To Notify

Don’t notify of successful operations, except in the following circumstances:

- **Security.** Users consider security operations to be of the highest importance, so notify users of successful security operations.
- **Recent failure.** Users don't take successful operations for granted if they were failing immediately before, so notify users of success when the operation was recently failing.
- **Prevent inconvenience.** Report successful operations when doing so might avoid inconveniencing users. Consequently, notify users when a successful operation is performed in an unexpected way, such as when an operation is lengthy or completes earlier or later than expected.
- **In other circumstances, either give no feedback for success or give feedback "on demand."** Assume that users take successful operations for granted. You can give feedback on demand by displaying an icon (or changing an existing icon) in the notification area while the operation is being performed, and removing the icon (or restoring the previous icon) when the operation is complete.
  - For the FYI pattern, don't give a notification if users can continue to work normally or are unlikely to do anything different as the result of the notification.
  - Exception: You can notify users of information of questionable relevance if it is optional and users opt in.
  - For the noncritical system event and FYI patterns, use complete notifications for a single event. Don't present several partial ones.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
393 / 867 100% Find

When to notify

- Display a notification based on its design pattern:

Pattern	When to notify
Action success	Upon completion of an asynchronous task. Notify users of successful actions only if they are likely to be waiting for completion, or after recent failures.
Action failure	Upon failure of an asynchronous task.
Non-critical system event	When an event occurs and the user is active, or the condition continues to exist. If this results from a problem, remove the currently displayed notification immediately once the problem is resolved.
Optional user task	When the need to perform a task is determined and the user is active.
FYI	When the triggering event occurs.

Figure 3-40: When to notify

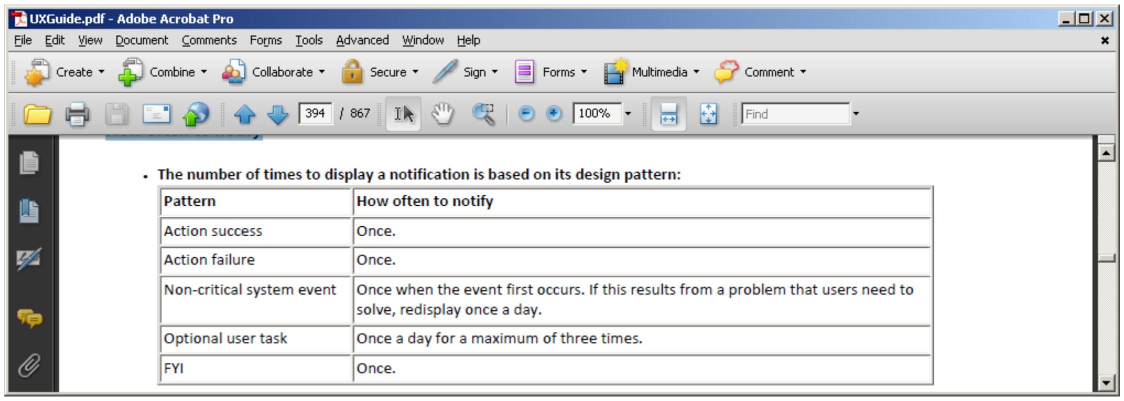
- For the action failure pattern, **if the problem might correct itself within seconds, delay the failure notification for an appropriate amount of time.** If the problem corrects itself, report nothing. Notify only after enough time has passed that the failure is noticeable. If you report too early, most likely users won't notice the problem reported, but they will notice the unnecessary notification.

- For the action success and FYI patterns, **use the real-time option so that stale notifications aren't queued** when users are running a full-screen application or aren't actively using their computer.
- For the non-critical system event pattern, **don't create the potential for notification storms by staggering events tied to well-known events such as user logon.** Instead, tie the event to some time period after the event. For example, you could remind users to register your product five minutes after user logon.

3.37.1.3 How Long To Notify

In Windows Vista and later, notifications are displayed for a fixed duration of 9 seconds.

3.37.1.4 How Often To Notify


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
394 / 867
100%
Find

- The number of times to display a notification is based on its design pattern:

Pattern	How often to notify
Action success	Once.
Action failure	Once.
Non-critical system event	Once when the event first occurs. If this results from a problem that users need to solve, redisplay once a day.
Optional user task	Once a day for a maximum of three times.
FYI	Once.

Figure 3-41: The number of times to display a notification is based on its design pattern

For optional user tasks, don't try to pester users into submission by constantly displaying notifications. If the task is required, display a modal dialog box immediately instead of using notifications.

3.37.1.5 Notification Escalation

Don't assume that users will see your notifications. Users won't see them when:

- They are immersed in their work.
- They aren't paying attention.
- They are away from their computer.
- They are running a full-screen application.
- Their administrator has turned off all notifications for their computer.

If users must eventually take some kind of action, use progressive escalation to display an alternative UI that users cannot ignore.

3.37.1.6 Interaction

Make notifications clickable when:

- **Users should perform an action.** Clicking the notification should display a window in which users can perform the action. This approach is preferred for the action failure and optional user task design patterns.
- **Users may want to see more information.** Clicking the notification should display a window in which users can view additional information.
- **Always display a window when users click to perform an action.** Don’t have clicking perform an action directly.
- **Clicking to show more information should always show more information.** Don’t just rephrase the information already in the notification.

3.37.1.7 Icons

- For the action failure pattern, use the standard error icon.
- For the noncritical system event patterns, use the standard warning icon.
- For other patterns, use icons showing objects that relate to or suggest the subject, such as a shield for security or a battery for power.
- Use icons based on your application or company branding if your target users will recognize them and there is no better alternative.
- For progressive escalation, consider using icons with a progressively more emphatic appearance as the situation becomes more urgent
- Don’t use the standard information icon. That notifications are information goes without saying.
  - **Consider using large icons (32 × 32 pixels) when:** Users will quickly comprehend the icon rather than the text.
  - The large icons convey their meaning more clearly and effectively than the standard 16 × 16 pixel icons.
  - The icon uses the **Aero-style**.

3.37.1.8 Notification Queuing

- **For the action success and FYI patterns, use the real-time option** so that the notification isn’t queued for long. These notifications have value only when they can be displayed immediately.
- **Remove queued notifications when they are no longer relevant. Developers:** You can do this by setting the NIF_INFO flag in uFlags and set szInfo to an empty string. There is no harm in doing this if the notification is no longer in the queue.

3.37.1.9 System Integration

If your application doesn't always have an icon in the notification area when it's running, display an icon temporarily during the asynchronous task or event that caused the notification.

3.37.2 Text

3.37.2.1 Title Text

- Use title text that briefly summarizes the most important information you need to communicate to users in clear, plain, concise, specific language. Users should be able to understand the purpose of the notification information quickly and with minimal effort.
- Use text fragments or complete sentences without ending punctuation.
- Use sentence-style capitalization.
- Use no more than 48 characters (in English) to accommodate localization. The title has a maximum length of 63 characters, but you must allow for 30% expansion when the English-language text is translated.

3.37.2.2 Body Text

- Use body text that gives a description (without repeating the information in the title) and, optionally, that gives specific details about the notification, and also lets users know what action is available.
- Use complete sentences with ending punctuation.
- Use sentence-style capitalization.
- Use no more than 200 characters (in English) to accommodate localization. The body text has a maximum length of 255 characters, but you must allow for 30 percent expansion when the English-language text is translated.
- Include essential information in the body text, such as specific object names. (Examples: user names, file names, or URLs.) Users :
  - Shouldn't have to open another window to find such information.
- Put double quotation marks around object names. Exception: Don't use quotation marks when:
  - The object name always uses title-style capitalization, such as with user names.
  - The object name is offset with a colon (example: Printer name: My printer).
  - The object name can be easily determined from the context.

If you must truncate object names to a fixed maximum size to accommodate localization, use an ellipsis to indicate truncation.

Use the following phrasing if the notification is actionable:

- If users can click the notification to perform an action: <brief description of essential information> <optional details> Click to <do something>.
- If users can click the notification to see more information: <brief description of essential information> <optional details> Click for more information.
- **Don't say that the user "must" perform an action in a notification.** Notifications are for non-critical information that users can freely ignore. If users really must perform an action, don't use notifications.
- **If users should perform an action, make the importance clear.**
- For the action failure and non-critical system event patterns, **describe problems in plain language.**
- **Describe the event in a way that is relevant to the target users.** A notification is relevant if there's a reasonable chance that users will perform a task or change their behavior as the result of the notification. You can often accomplish this by describing notifications in terms of user goals instead of technological issues.

3.38 Interaction (Page 398)

3.38.1 Guidelines

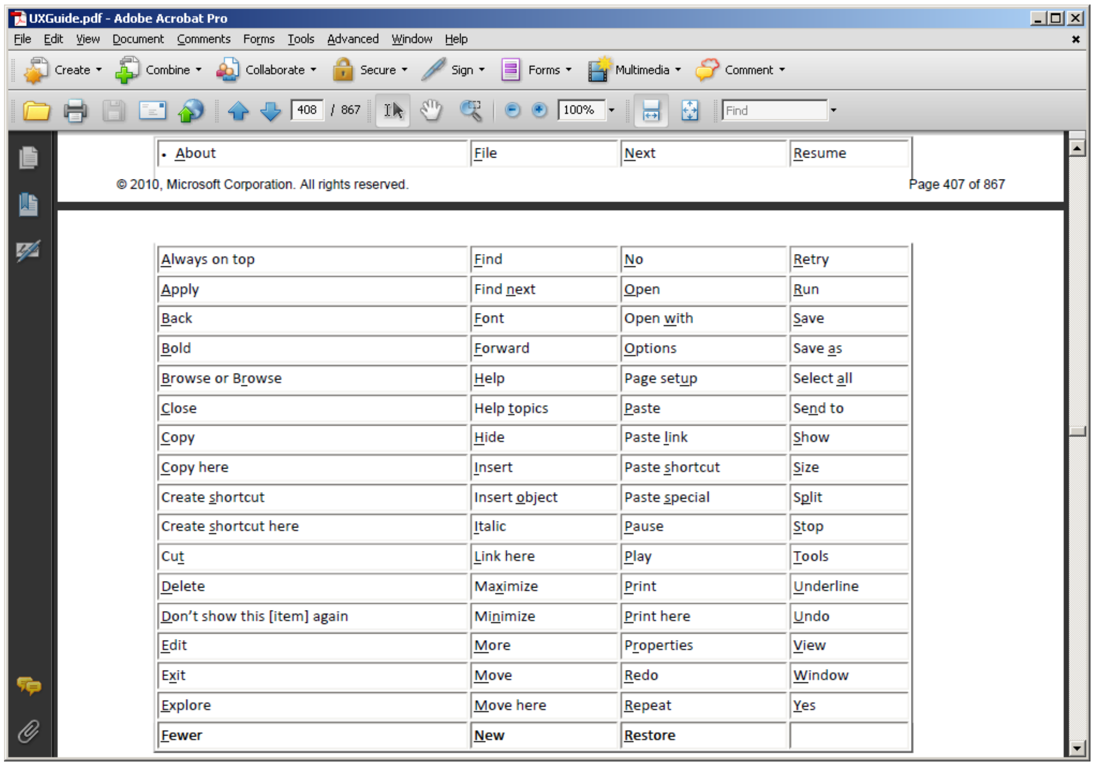
3.38.1.1 Interaction

- Don't use the Shift key to modify commands in menus or dialog boxes. Doing so is undiscoverable and unexpected. Incorrect:
- Don't disable a control with input focus. Doing so may prevent the window from receiving keyboard input. Instead, before disabling a control with input focus, move input focus to another control.
- If a window is displayed out of context, potentially surprising users, you may need to prevent significant unintended consequences:
  – Don't assign a default button.
  – Don't assign access keys.
  – Give initial input focus to a control other than a commit button.

3.38.1.2 Keyboard Navigation

- **Always show the input focus indicator. Exception:** You can temporarily suppress the input focus indicator if: The input focus indicator is visually distracting (as with a large list view not in Details view).
- The Enter key's usage is likely preceded by other keyboard input, such as Alt or arrow keys.
- The input focus indicator is displayed upon any keyboard input.
- **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 follow reading order, which generally flows from left to right, top to bottom.** 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 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.
- **Select the safest (to prevent loss of data or system access) and most secure command button or command link to be the default.** If safety and security aren't factors, select the most likely or convenient response.
- **Keyboard navigation shouldn't change control values or result in an error message.** Never require users to change a control's initial value during navigation. Instead, initialize controls that validate on exit with valid values, and validate a control's value only when it has changed.

3.38.1.3 Access Keys


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
408 / 867
100%
Find

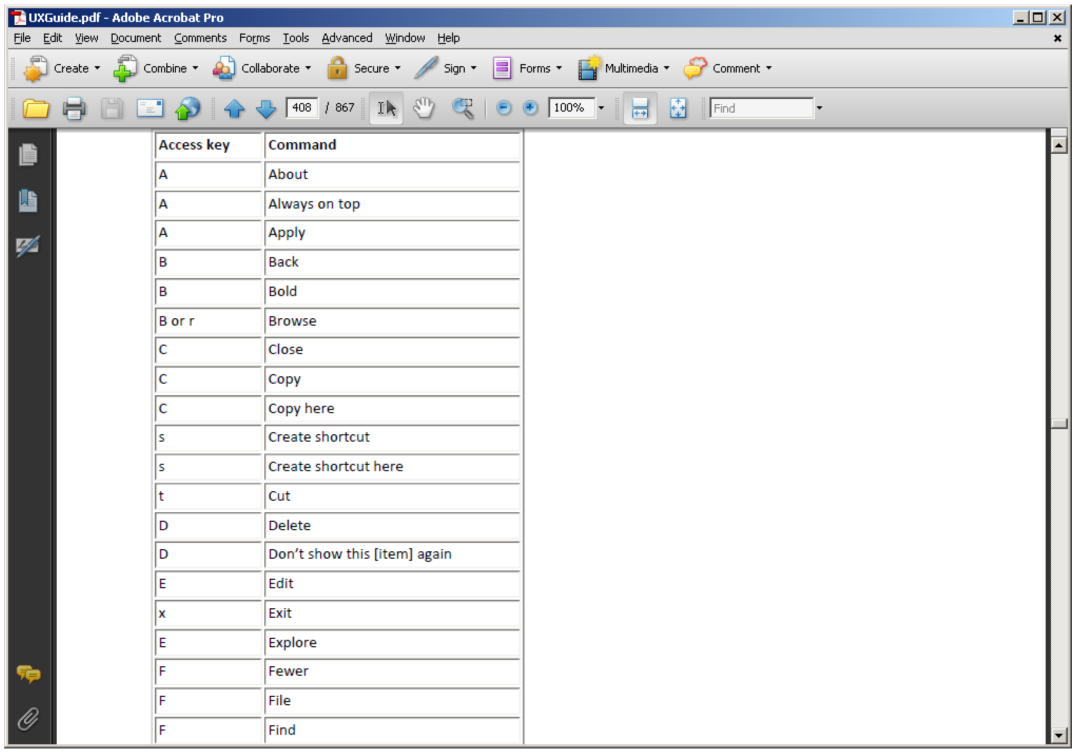
• About	File	Next	Resume

© 2010, Microsoft Corporation. All rights reserved. Page 407 of 867

Always on top	Find	No	Retry
Apply	Find next	Open	Run
Back	Font	Open with	Save
Bold	Forward	Options	Save as
Browse or Browse	Help	Page setup	Select all
Close	Help topics	Paste	Send to
Copy	Hide	Paste link	Show
Copy here	Insert	Paste shortcut	Size
Create shortcut	Insert object	Paste special	Split
Create shortcut here	Italic	Pause	Stop
Cut	Link here	Play	Tools
Delete	Maximize	Print	Underline
Don't show this [item] again	Minimize	Print here	Undo
Edit	More	Properties	View
Exit	Move	Redo	Window
Explore	Move here	Repeat	Yes
Fewer	New	Restore	

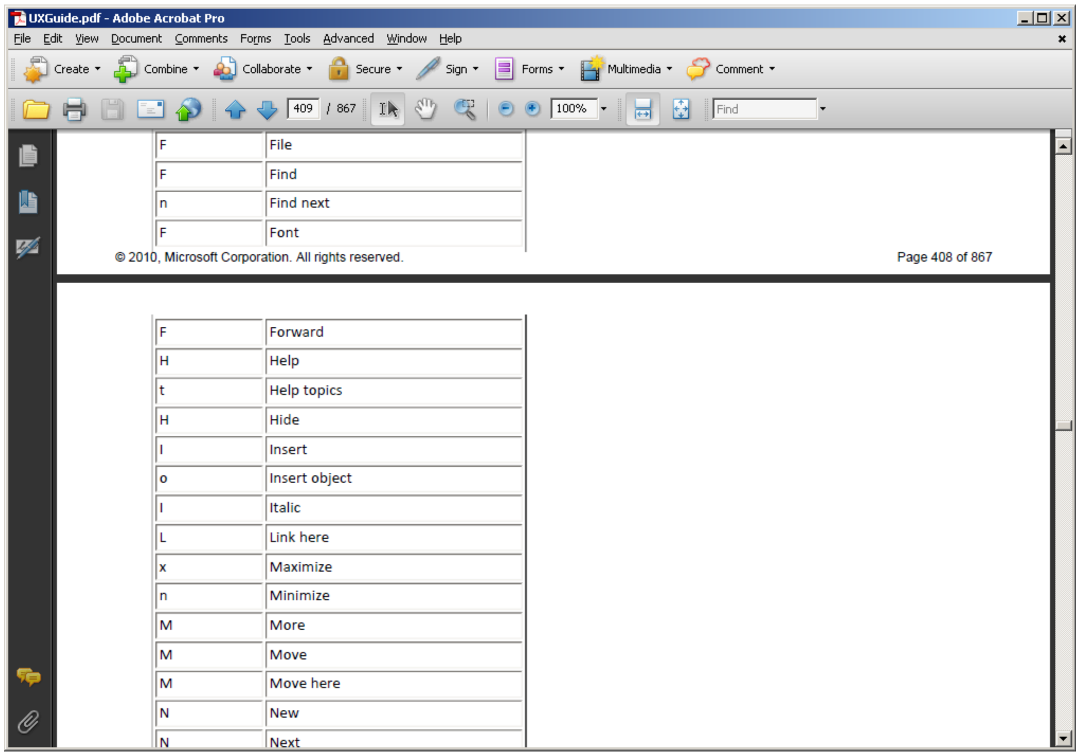
Figure 3-42: Access key assignments

Whenever possible, assign access keys for commonly used commands according to the following table. While consistent access key assignments aren't always possible, they are certainly preferred—especially for frequently used commands.



Access key	Command
A	About
A	Always on top
A	Apply
B	Back
B	Bold
B or r	Browse
C	Close
C	Copy
C	Copy here
s	Create shortcut
s	Create shortcut here
t	Cut
D	Delete
D	Don't show this [item] again
E	Edit
x	Exit
E	Explore
F	Fewer
F	File
F	Find

Figure 3-43: Access keys and commands

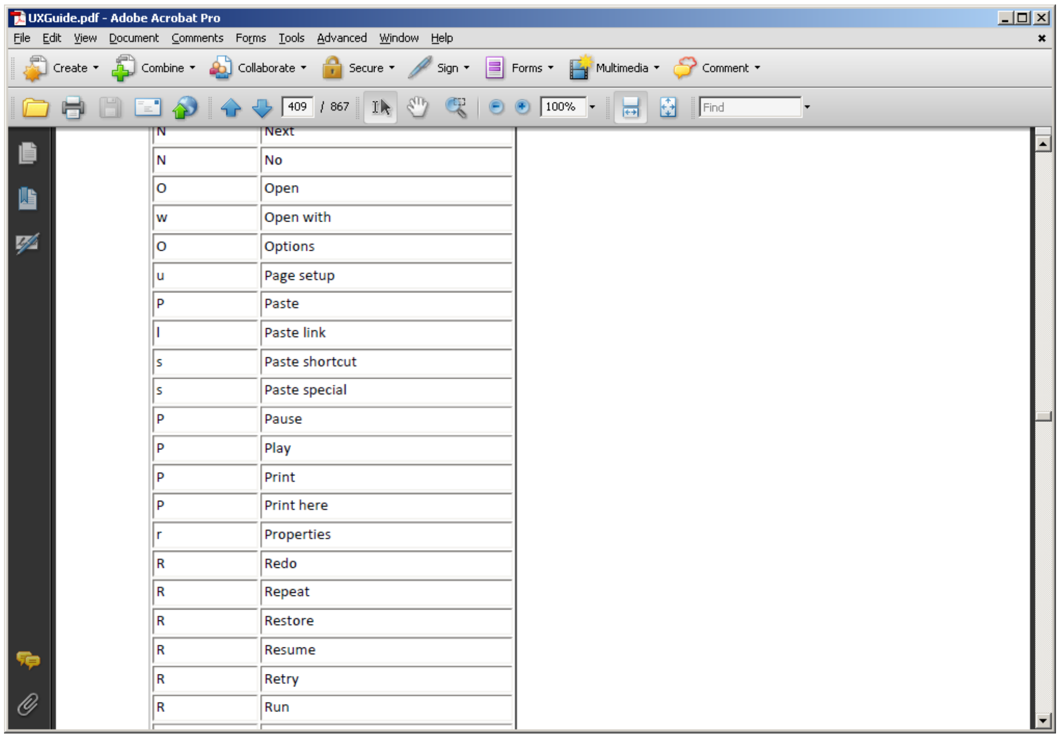


F	File
F	Find
n	Find next
F	Font

© 2010, Microsoft Corporation. All rights reserved. Page 408 of 867

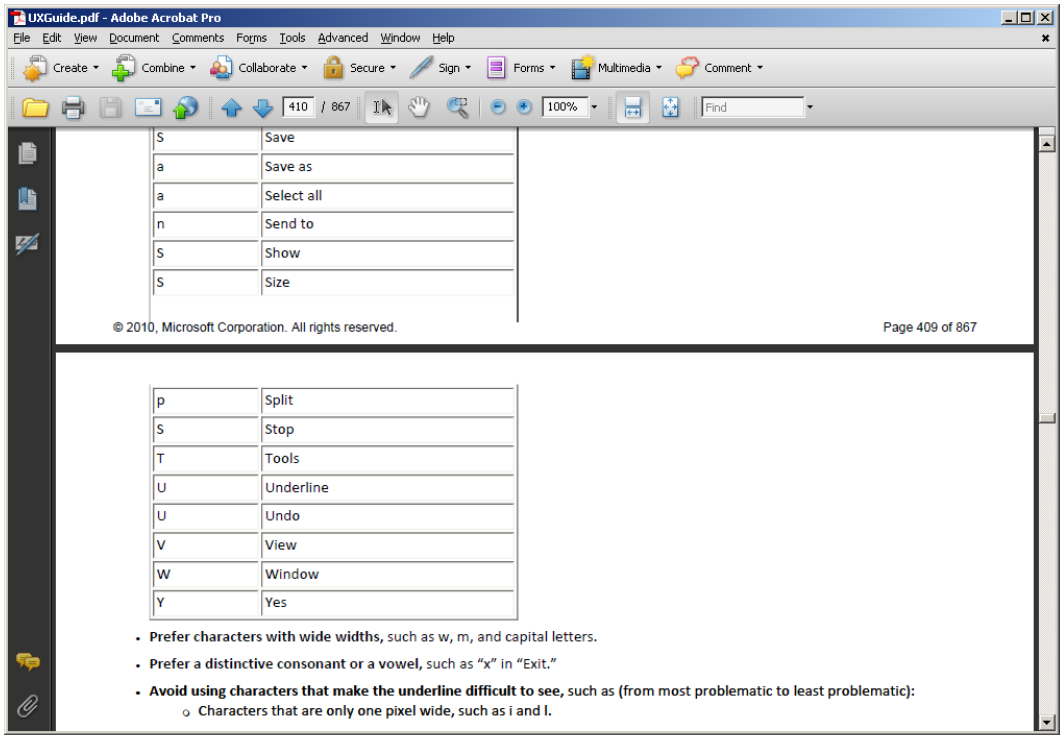
F	Forward
H	Help
t	Help topics
H	Hide
I	Insert
o	Insert object
I	Italic
L	Link here
x	Maximize
n	Minimize
M	More
M	Move
M	Move here
N	New
N	Next

Figure 3-44: Access keys and commands


UXGuide.pdf - Adobe Acrobat Pro

N	Next
N	No
O	Open
w	Open with
O	Options
u	Page setup
P	Paste
l	Paste link
s	Paste shortcut
s	Paste special
P	Pause
P	Play
P	Print
P	Print here
r	Properties
R	Redo
R	Repeat
R	Restore
R	Resume
R	Retry
R	Run

Figure 3-45: Access keys and commands


UXGuide.pdf - Adobe Acrobat Pro

S	Save
a	Save as
a	Select all
n	Send to
S	Show
S	Size

© 2010, Microsoft Corporation. All rights reserved. Page 409 of 867

p	Split
S	Stop
T	Tools
U	Underline
U	Undo
V	View
W	Window
Y	Yes

- **Prefer characters with wide widths**, such as w, m, and capital letters.
- **Prefer a distinctive consonant or a vowel**, such as "x" in "Exit."
- **Avoid using characters that make the underline difficult to see**, such as (from most problematic to least problematic):
  - Characters that are only one pixel wide, such as i and l.

Figure 3-46: Access keys and commands

- Prefer characters with wide widths, such as w, m, and capital letters.
- Prefer a distinctive consonant or a vowel, such as "x" in "Exit."

- Avoid using characters that make the underline difficult to see, such as (from most problematic to least problematic):
  - Characters that are only one pixel wide, such as “i” and “l”.
  - Characters with descenders, such as “g”, “j”, “p”, “q”, and “y”.
  - Characters next to a letter with a descender.
  - When assigning access keys in wizard pages, remember to reserve “B” for Back and “N” for Next.
  - When assigning access keys in property pages, remember to reserve “A” for Apply, if used.

3.38.1.4 Menu Access Keys

- Assign access keys to all menu items. No exceptions.
- For dynamic menu items (such as recently used files), assign access keys numerically.
- Assign unique access keys within a menu level. You can reuse access keys across different menu levels.
- Make access keys easy to find:
  - For the most frequently used menu items, choose characters at the beginning of the first or second word of the label, preferably the first character.
  - For less frequently used menu items, choose letters that are a distinctive consonant or a vowel in the label.

3.38.1.5 Dialog Box Access Keys

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.

- **Group labels.** Normally, the individual controls within a group are assigned access keys, so the group label doesn’t need one. However, assign an access key to the group label and not the individual controls if there is a shortage of access keys.
- **Generic Help buttons,** which are accessed with F1.
- **Link labels.** There are often too many links to assign unique access keys, and link underscores hide the access key underscores. Have users access links with the Tab key instead.
- **Tab names.** Tabs are cycled using Ctrl+Tab and Ctrl+Shift+Tab.

- **Browse buttons labeled "...".** These can't be assigned access keys uniquely.
- **Unlabeled controls,** such as spin controls, graphic command buttons, and unlabeled progressive disclosure controls.
- **Non-label static text or labels for controls that aren't interactive,** such as progress bars.
- **Assign commit button access keys first to ensure that they have the standard key assignments.** If there isn't a standard key assignment, use the first letter of the first word. For example, the access key for Yes and No commit buttons should always be "Y" and "N", regardless of the other controls in the dialog box.
- **For negative commit buttons (other than Cancel) phrased as a "Don't", assign the access key to the "n" in "Don't".** If not phrased as a "Don't", use the standard access key assignment or assign the first letter of the first word. By doing so, all Don'ts and No's have a consistent access key.
- **To make access keys easy to find, assign the access keys to a character that appears early in the label,** ideally the first character, even if there is a keyword that appears later in the label.
- **Assign at most 20 access keys,** so you have a few unassigned characters to facilitate localization.
- **If there are too many interactive controls to assign unique access keys, you may assign nonunique access keys** if: The controls would otherwise be too difficult to navigate to.
- The non-unique access keys don't conflict with the access keys of commonly used controls.
- **Don't use menu bars in dialog boxes.** It's difficult to assign unique access keys in this case, because the dialog box controls and menu items share the same characters.

3.38.1.6 Shortcut Keys

- **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. For the well-known shortcut keys used by Windows programs, see **Windows Keyboard Shortcut Keys**.
- **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.

- **Document all shortcut keys.** Document shortcuts in menu bar items, toolbar tooltips, and a single Help article that documents all shortcut keys used. Doing so helps users learn the shortcut key assignments—they shouldn't be a secret. **Exception:** Don't display shortcut key assignments within context menus. Context menus don't display the shortcut key assignments because these menus are optimized for efficiency.
- **If your program assigns many shortcut keys, provide the ability to customize the assignments.** Doing so allows users to reassign conflicting shortcut keys and migrate from other products. Most programs don't assign enough shortcut keys to need this feature.

3.38.1.7 Choosing Shortcut Keys

- **For well-known shortcut keys, use the standard assignments.** For the well-known shortcut keys used by Windows programs, see **Windows Keyboard Shortcut Keys**.
- **For non-standard key assignments, use the following recommended shortcut keys for more frequently used commands.** These shortcut keys are recommended because they don't conflict with the well-known shortcuts and are easy to press:
  - Ctrl-G, J, K, L M, Q, R, or T
  - Ctrl-any number
  - F7, F8, F9, or F12
  - Shift-F2, F3, F4, F5, F7, F8, F9, F11, or F12
  - Alt-any function key except F4
- **Use the following recommended shortcut keys for less frequently used commands.** These shortcut keys don't have conflicts, but are harder to press—often requiring two hands.
  - Ctrl-any function key except F4 and F6
  - Ctrl+-Shift-any letter or number
- Make frequently used shortcut keys easy to remember:
  - Use letters instead of numbers or function keys.
  - Try to use a letter that is in the first word or most memorable character within the command's keywords.
- Use function keys for commands that have a small-scale effect, such as commands that apply to the selected object. For example, F2 renames the selected item.
- Use Ctrl key combinations for commands that have a large-scale effect, such as commands that apply to an entire document. For example, Ctrl-S saves the current document.

- Use Shift key combinations for commands that extend or complement the actions of the standard shortcut key. For example, the Al-Tab shortcut key cycles through open primary windows, whereas Alt-Shift-Tab cycles in the reverse order. Similarly, F1 displays Help, whereas Shift-F1 displays context-sensitive Help.
- When using arrow keys to move or resize an item, use Ctrl-arrow keys for more granular control.

3.38.1.8 Choosing Shortcut Keys (What Not To Do)

- **Don't distinguish between key locations.** For example, Windows can distinguish between left and right Shift, Alt, Ctrl, **Windows logo**, and **Application keys**, as well as keys on the numeric keypad. Assigning behavior to only one key location is confusing and unexpected.
- **Don't use the Windows logo modifier key for program shortcut keys.** Windows logo key is reserved for Windows use. Even if a Windows logo key combination isn't being used by Windows now, it may be in the future.
- **Don't use the Application key as a shortcut key modifier.** Use Ctrl, Alt, and Shift instead.
- **Don't use shortcut keys used by Windows for program shortcut keys.** Doing so will conflict with the Windows system shortcut keys when your program has input focus. For the shortcut keys used by Windows, see **Windows Keyboard Shortcut Keys**.
- **Don't use Alt-alphanumeric key combinations for shortcut keys.** Such shortcut keys may conflict with access keys.
- **Don't use the following characters for shortcut keys:** @ £ $ {} [] \ ~ | ^ ' < >. These characters require different key combinations across languages or are locale specific.
- **Avoid complex key combinations,** such as three or more keys together (example: Ctrl-Alt-spacebar) or keys that are far apart on the keyboard (example: Ctrl-F5). Use simple shortcut keys for frequently used commands.
- **Don't use Ctrl-Alt combinations,** because Windows interprets this combination in some language versions as an AltGR key, which generates alphanumeric characters.

3.38.2 Keyboard And Mouse Combinations

For links, use Shift-click to navigate using a new window and Ctrl-click to navigate using a new tab. This approach is consistent with Windows Internet Explorer®.

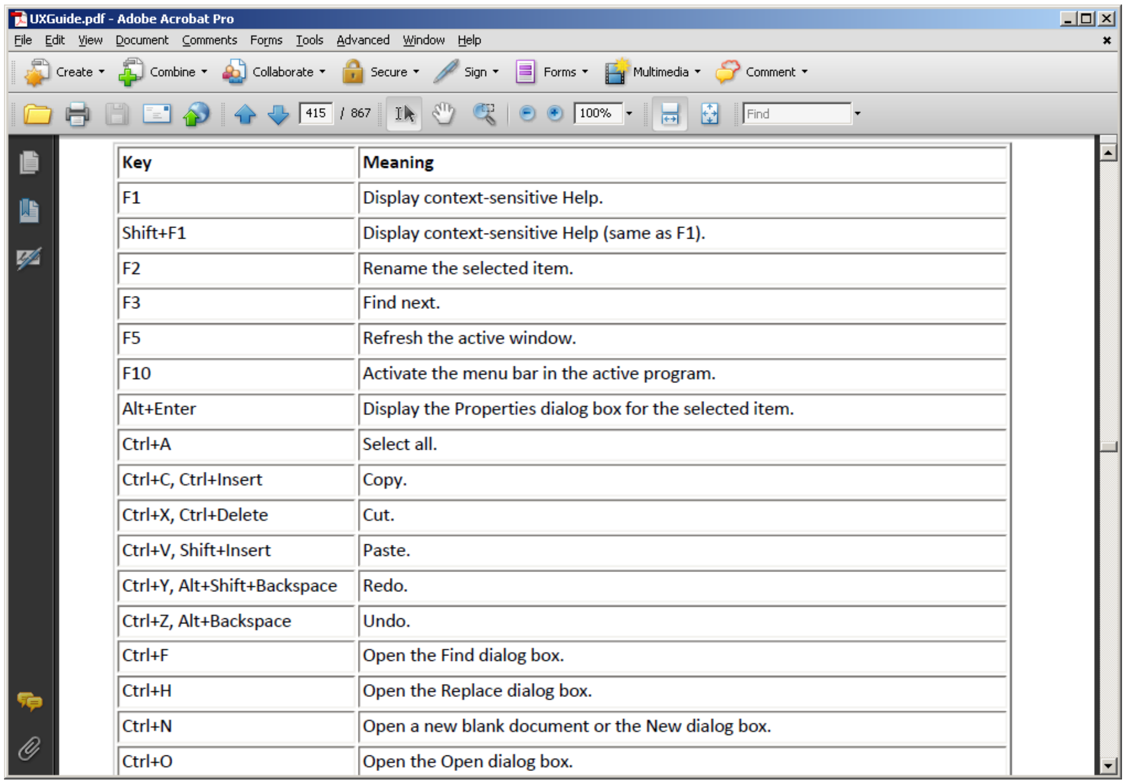
3.39 Windows Keyboard Shortcut Keys (Page 415)

3.39.1 Keyboard

The following tables summarize the standard Microsoft® Windows® shortcut key assignments.

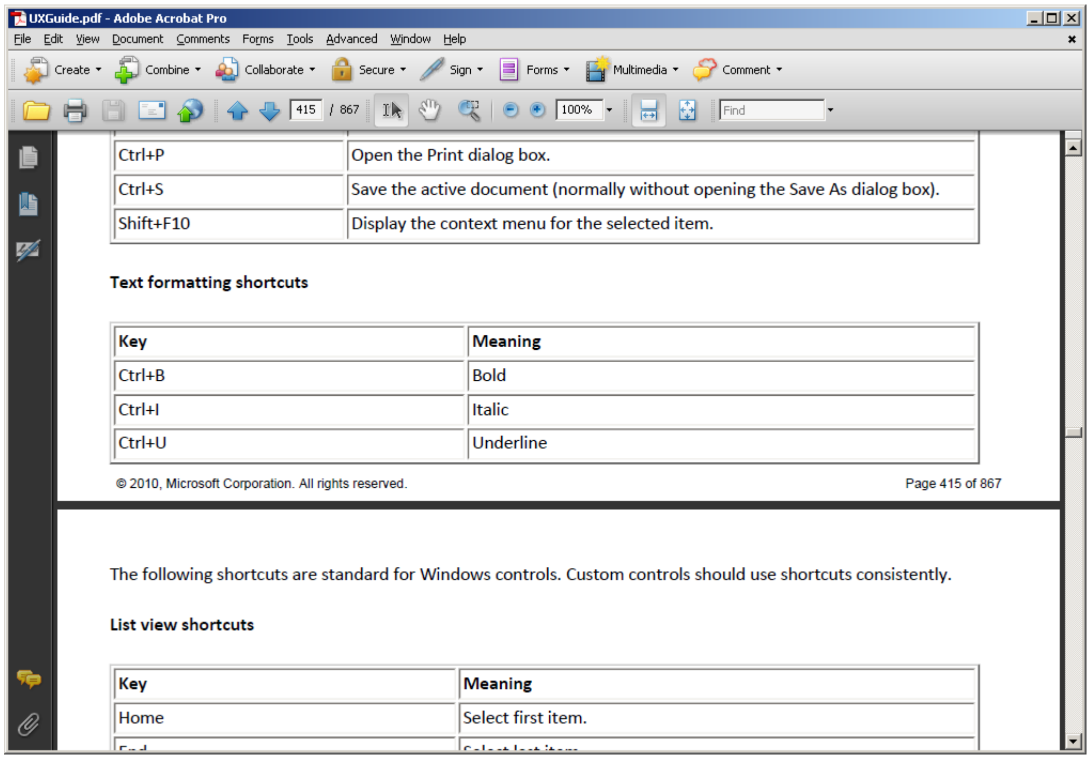
The following shortcuts can be used by any program, but they must have the given meaning.

3.39.1.1 General Shortcuts


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
415 / 867
100%
Find

Key	Meaning
F1	Display context-sensitive Help.
Shift+F1	Display context-sensitive Help (same as F1).
F2	Rename the selected item.
F3	Find next.
F5	Refresh the active window.
F10	Activate the menu bar in the active program.
Alt+Enter	Display the Properties dialog box for the selected item.
Ctrl+A	Select all.
Ctrl+C, Ctrl+Insert	Copy.
Ctrl+X, Ctrl+Delete	Cut.
Ctrl+V, Shift+Insert	Paste.
Ctrl+Y, Alt+Shift+Backspace	Redo.
Ctrl+Z, Alt+Backspace	Undo.
Ctrl+F	Open the Find dialog box.
Ctrl+H	Open the Replace dialog box.
Ctrl+N	Open a new blank document or the New dialog box.
Ctrl+O	Open the Open dialog box.

Figure 3-47: Shortcuts


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
415 / 867
100%
Find

Ctrl+P	Open the Print dialog box.
Ctrl+S	Save the active document (normally without opening the Save As dialog box).
Shift+F10	Display the context menu for the selected item.

Text formatting shortcuts

Key	Meaning
Ctrl+B	Bold
Ctrl+I	Italic
Ctrl+U	Underline

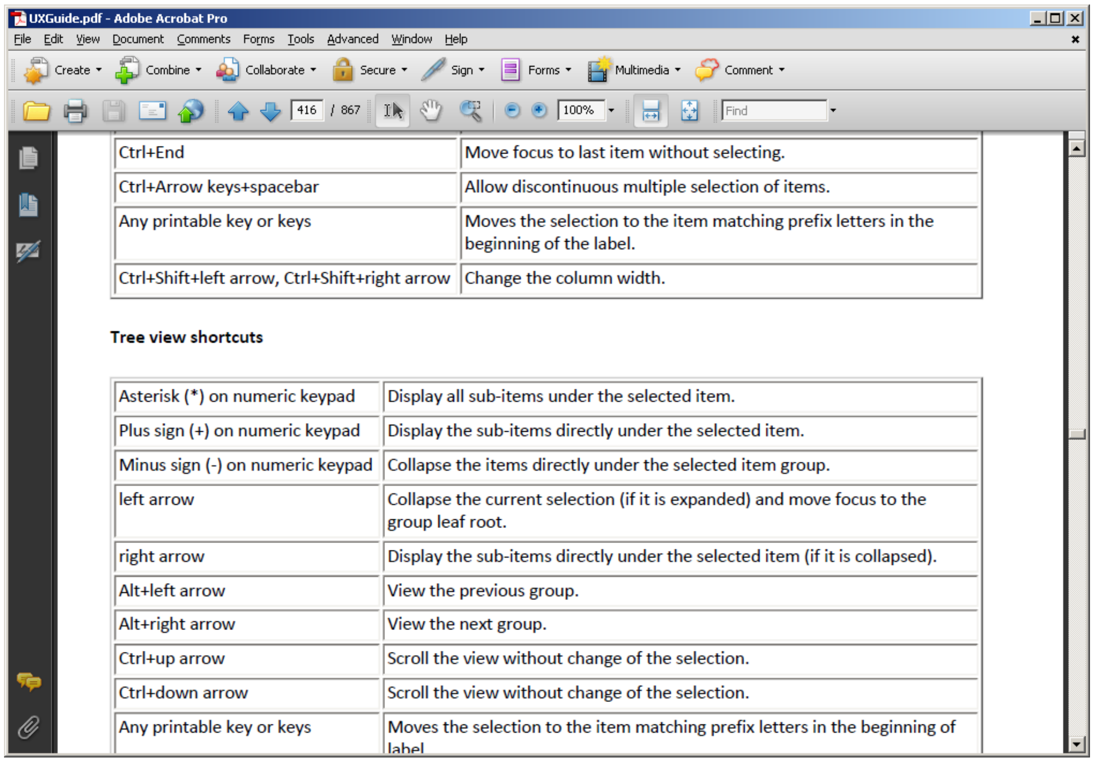
© 2010, Microsoft Corporation. All rights reserved. Page 415 of 867

The following shortcuts are standard for Windows controls. Custom controls should use shortcuts consistently.

List view shortcuts

Key	Meaning
Home	Select first item.

Figure 3-48: Text formatting shortcuts

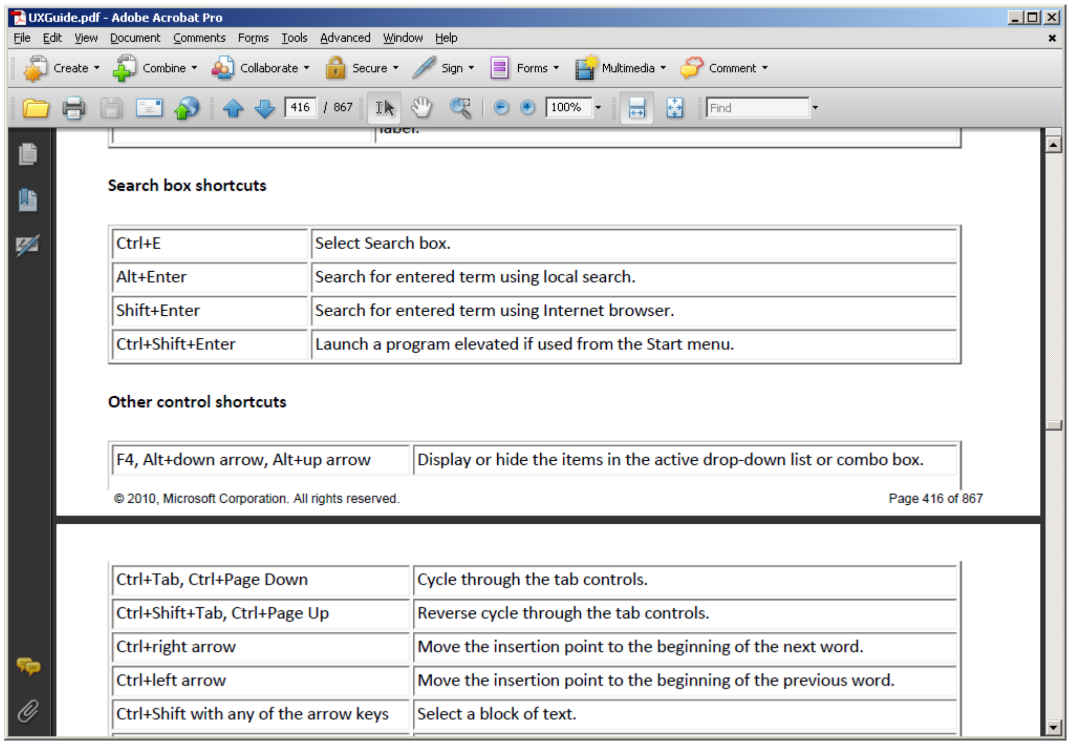

UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
416 / 867
100%
Find

Ctrl+End	Move focus to last item without selecting.
Ctrl+Arrow keys+spacebar	Allow discontinuous multiple selection of items.
Any printable key or keys	Moves the selection to the item matching prefix letters in the beginning of the label.
Ctrl+Shift+left arrow, Ctrl+Shift+right arrow	Change the column width.

Tree view shortcuts

Asterisk (*) on numeric keypad	Display all sub-items under the selected item.
Plus sign (+) on numeric keypad	Display the sub-items directly under the selected item.
Minus sign (-) on numeric keypad	Collapse the items directly under the selected item group.
left arrow	Collapse the current selection (if it is expanded) and move focus to the group leaf root.
right arrow	Display the sub-items directly under the selected item (if it is collapsed).
Alt+left arrow	View the previous group.
Alt+right arrow	View the next group.
Ctrl+up arrow	Scroll the view without change of the selection.
Ctrl+down arrow	Scroll the view without change of the selection.
Any printable key or keys	Moves the selection to the item matching prefix letters in the beginning of label

Figure 3-49: Tree view shortcuts


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
416 / 867
100%
Find

Search box shortcuts

Ctrl+E	Select Search box.
Alt+Enter	Search for entered term using local search.
Shift+Enter	Search for entered term using Internet browser.
Ctrl+Shift+Enter	Launch a program elevated if used from the Start menu.

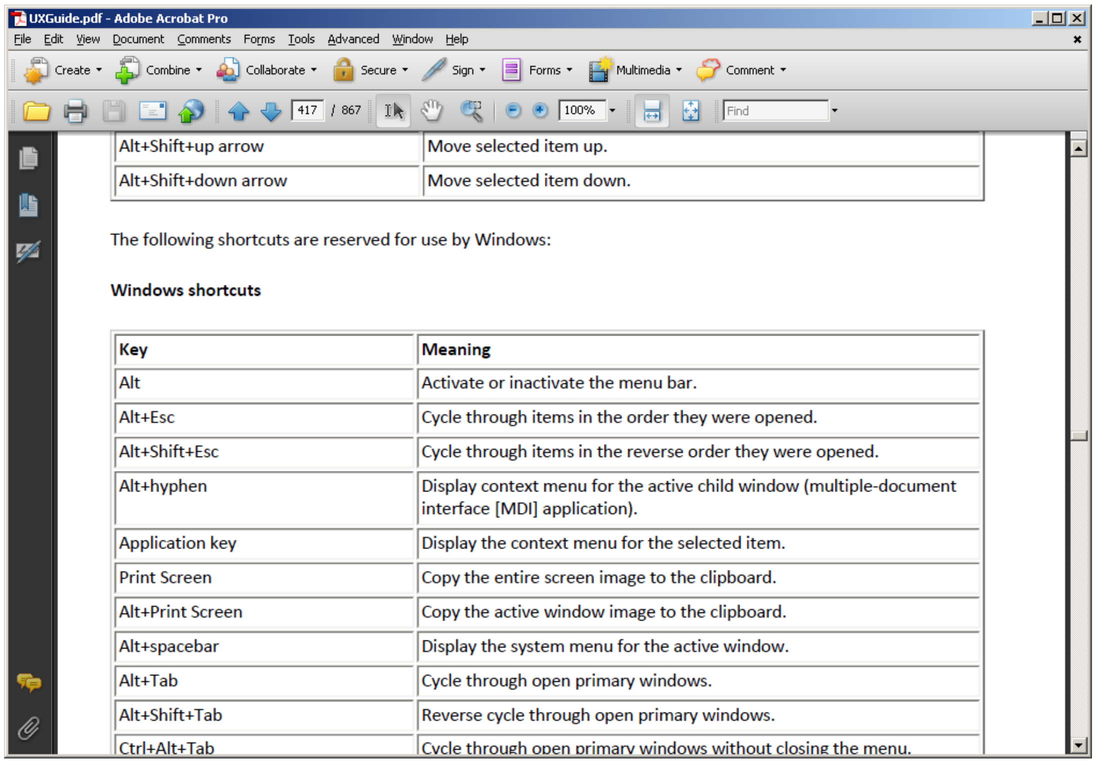
Other control shortcuts

F4, Alt+down arrow, Alt+up arrow	Display or hide the items in the active drop-down list or combo box.

© 2010, Microsoft Corporation. All rights reserved. Page 416 of 867

Ctrl+Tab, Ctrl+Page Down	Cycle through the tab controls.
Ctrl+Shift+Tab, Ctrl+Page Up	Reverse cycle through the tab controls.
Ctrl+right arrow	Move the insertion point to the beginning of the next word.
Ctrl+left arrow	Move the insertion point to the beginning of the previous word.
Ctrl+Shift with any of the arrow keys	Select a block of text.

Figure 3-50: Search box and other control shortcuts


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
417 / 867
100%
Find

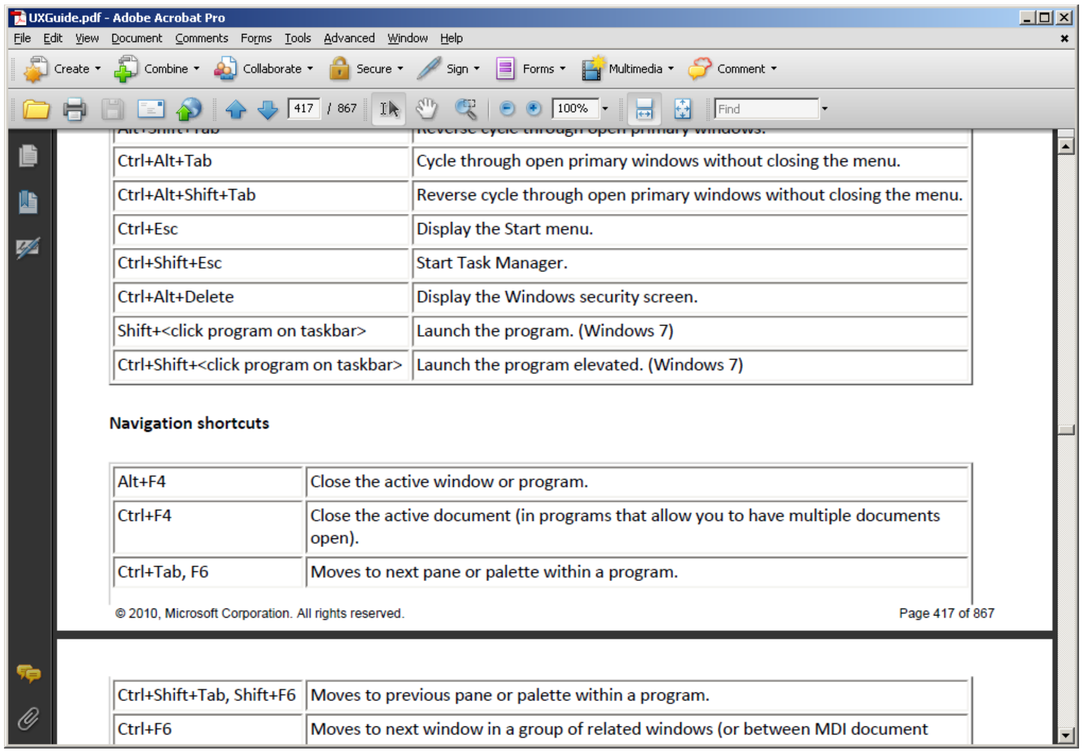
Alt+Shift+up arrow	Move selected item up.
Alt+Shift+down arrow	Move selected item down.

The following shortcuts are reserved for use by Windows:

Windows shortcuts

Key	**Meaning**
Alt	Activate or inactivate the menu bar.
Alt+Esc	Cycle through items in the order they were opened.
Alt+Shift+Esc	Cycle through items in the reverse order they were opened.
Alt+hyphen	Display context menu for the active child window (multiple-document interface [MDI] application).
Application key	Display the context menu for the selected item.
Print Screen	Copy the entire screen image to the clipboard.
Alt+Print Screen	Copy the active window image to the clipboard.
Alt+spacebar	Display the system menu for the active window.
Alt+Tab	Cycle through open primary windows.
Alt+Shift+Tab	Reverse cycle through open primary windows.
Ctrl+Alt+Tab	Cycle through open primary windows without closing the menu.

Figure 3-51: Windows shortcuts


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
417 / 867 100% Find

Ctrl+Alt+Tab	Cycle through open primary windows without closing the menu.
Ctrl+Alt+Shift+Tab	Reverse cycle through open primary windows without closing the menu.
Ctrl+Esc	Display the Start menu.
Ctrl+Shift+Esc	Start Task Manager.
Ctrl+Alt+Delete	Display the Windows security screen.
Shift+<click program on taskbar>	Launch the program. (Windows 7)
Ctrl+Shift+<click program on taskbar>	Launch the program elevated. (Windows 7)

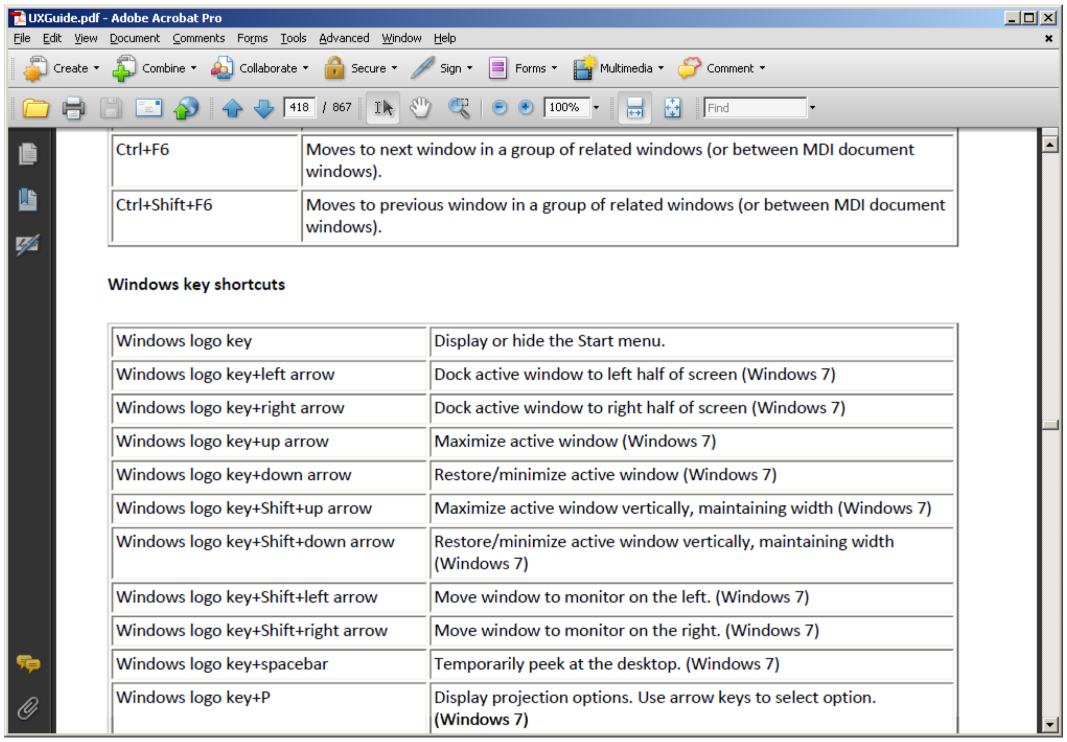
Navigation shortcuts

Alt+F4	Close the active window or program.
Ctrl+F4	Close the active document (in programs that allow you to have multiple documents open).
Ctrl+Tab, F6	Moves to next pane or palette within a program.

© 2010, Microsoft Corporation. All rights reserved. Page 417 of 867

Ctrl+Shift+Tab, Shift+F6	Moves to previous pane or palette within a program.
Ctrl+F6	Moves to next window in a group of related windows (or between MDI document

Figure 3-52: Navigation shortcuts

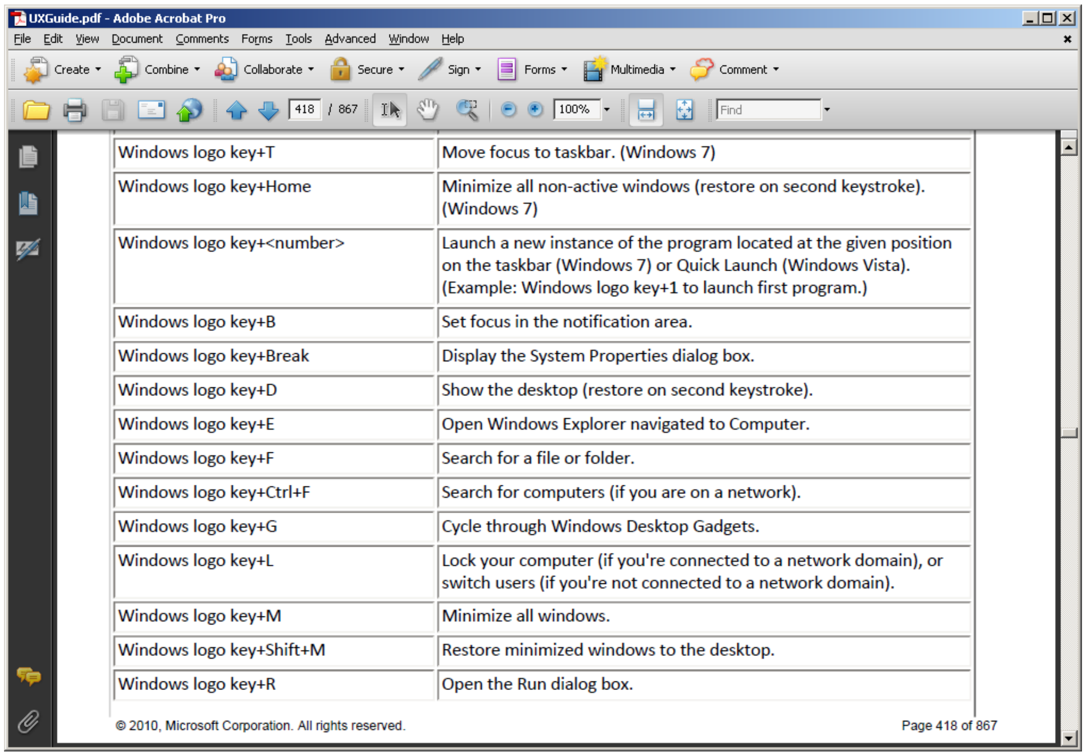

UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
418 / 867 100% Find

Ctrl+F6	Moves to next window in a group of related windows (or between MDI document windows).
Ctrl+Shift+F6	Moves to previous window in a group of related windows (or between MDI document windows).

Windows key shortcuts

Windows logo key	Display or hide the Start menu.
Windows logo key+left arrow	Dock active window to left half of screen (Windows 7)
Windows logo key+right arrow	Dock active window to right half of screen (Windows 7)
Windows logo key+up arrow	Maximize active window (Windows 7)
Windows logo key+down arrow	Restore/minimize active window (Windows 7)
Windows logo key+Shift+up arrow	Maximize active window vertically, maintaining width (Windows 7)
Windows logo key+Shift+down arrow	Restore/minimize active window vertically, maintaining width (Windows 7)
Windows logo key+Shift+left arrow	Move window to monitor on the left. (Windows 7)
Windows logo key+Shift+right arrow	Move window to monitor on the right. (Windows 7)
Windows logo key+spacebar	Temporarily peek at the desktop. (Windows 7)
Windows logo key+P	Display projection options. Use arrow keys to select option. (Windows 7)

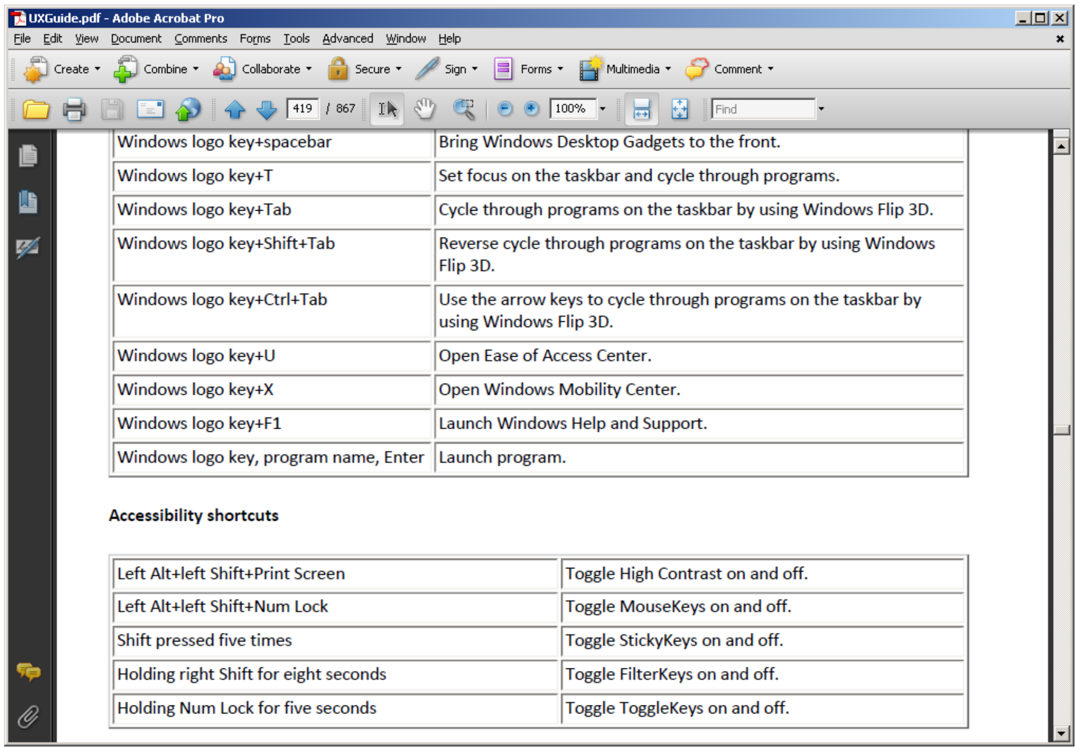
Figure 3-53: Windows key shortcuts


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
418 / 867 100% Find

Windows logo key+T	Move focus to taskbar. (Windows 7)
Windows logo key+Home	Minimize all non-active windows (restore on second keystroke). (Windows 7)
Windows logo key+<number>	Launch a new instance of the program located at the given position on the taskbar (Windows 7) or Quick Launch (Windows Vista). (Example: Windows logo key+1 to launch first program.)
Windows logo key+B	Set focus in the notification area.
Windows logo key+Break	Display the System Properties dialog box.
Windows logo key+D	Show the desktop (restore on second keystroke).
Windows logo key+E	Open Windows Explorer navigated to Computer.
Windows logo key+F	Search for a file or folder.
Windows logo key+Ctrl+F	Search for computers (if you are on a network).
Windows logo key+G	Cycle through Windows Desktop Gadgets.
Windows logo key+L	Lock your computer (if you're connected to a network domain), or switch users (if you're not connected to a network domain).
Windows logo key+M	Minimize all windows.
Windows logo key+Shift+M	Restore minimized windows to the desktop.
Windows logo key+R	Open the Run dialog box.

© 2010, Microsoft Corporation. All rights reserved. Page 418 of 867

Figure 3-54: Windows key shortcuts

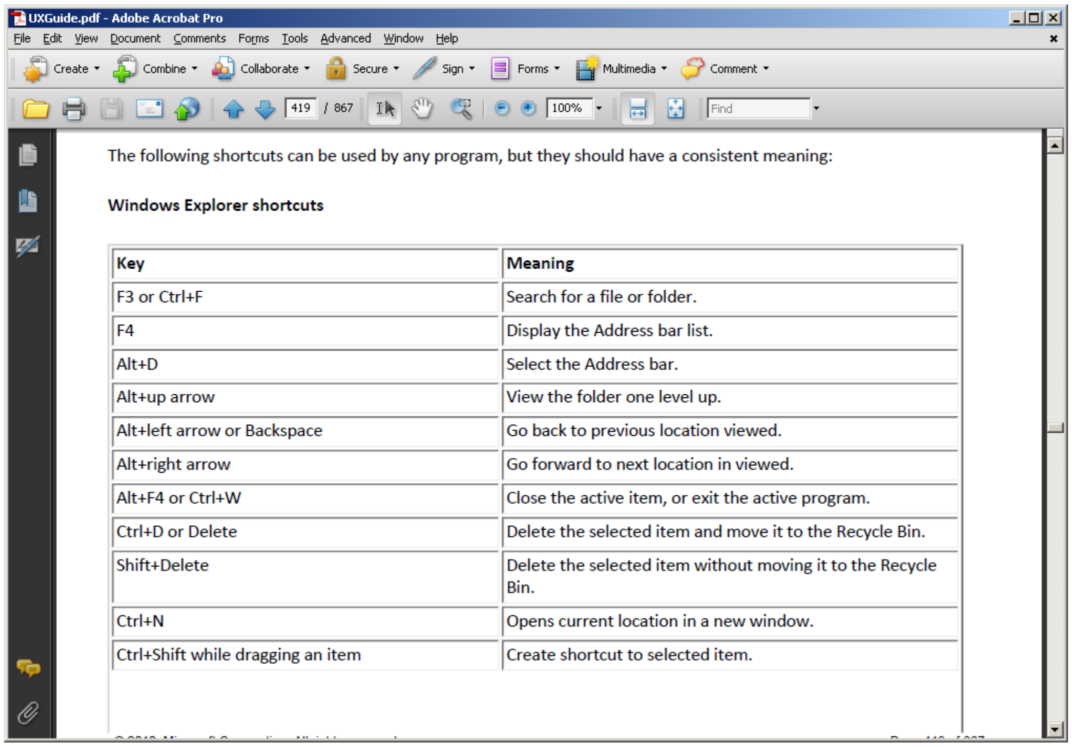

UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
419 / 867 100% Find

Windows logo key+spacebar	Bring Windows Desktop Gadgets to the front.
Windows logo key+T	Set focus on the taskbar and cycle through programs.
Windows logo key+Tab	Cycle through programs on the taskbar by using Windows Flip 3D.
Windows logo key+Shift+Tab	Reverse cycle through programs on the taskbar by using Windows Flip 3D.
Windows logo key+Ctrl+Tab	Use the arrow keys to cycle through programs on the taskbar by using Windows Flip 3D.
Windows logo key+U	Open Ease of Access Center.
Windows logo key+X	Open Windows Mobility Center.
Windows logo key+F1	Launch Windows Help and Support.
Windows logo key, program name, Enter	Launch program.

Accessibility shortcuts

Left Alt+left Shift+Print Screen	Toggle High Contrast on and off.
Left Alt+left Shift+Num Lock	Toggle MouseKeys on and off.
Shift pressed five times	Toggle StickyKeys on and off.
Holding right Shift for eight seconds	Toggle FilterKeys on and off.
Holding Num Lock for five seconds	Toggle ToggleKeys on and off.

Figure 3-55: Accessibility shortcuts

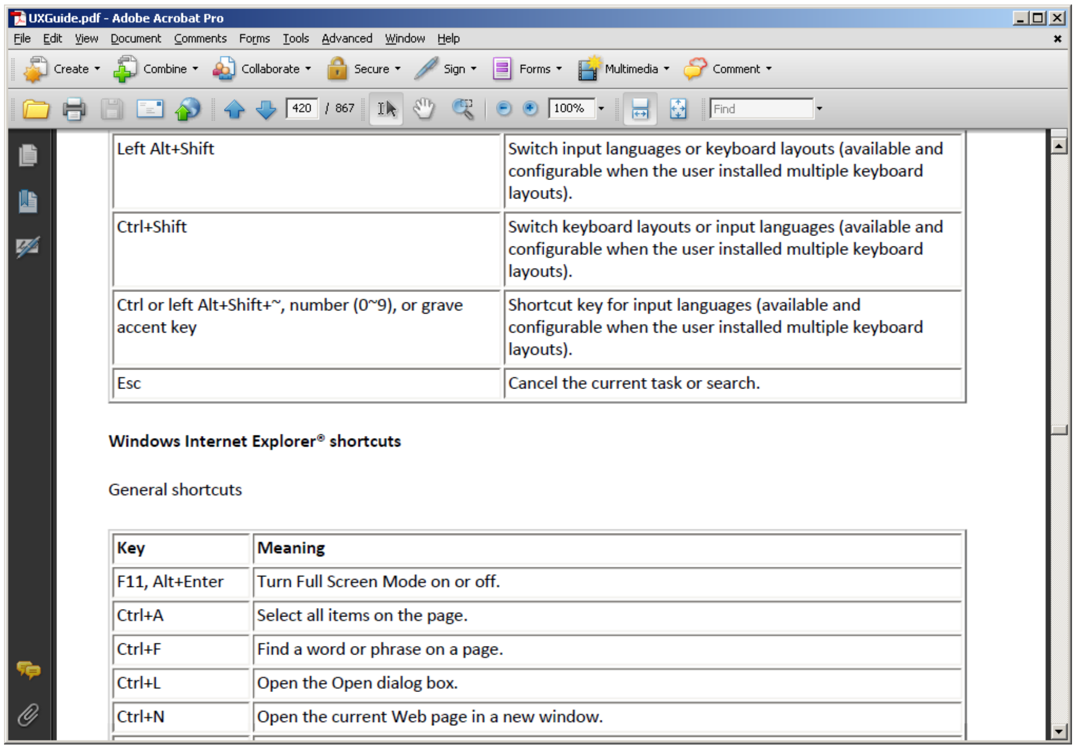

UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
419 / 867
100%
Find

The following shortcuts can be used by any program, but they should have a consistent meaning:

Windows Explorer shortcuts

Key	Meaning
F3 or Ctrl+F	Search for a file or folder.
F4	Display the Address bar list.
Alt+D	Select the Address bar.
Alt+up arrow	View the folder one level up.
Alt+left arrow or Backspace	Go back to previous location viewed.
Alt+right arrow	Go forward to next location in viewed.
Alt+F4 or Ctrl+W	Close the active item, or exit the active program.
Ctrl+D or Delete	Delete the selected item and move it to the Recycle Bin.
Shift+Delete	Delete the selected item without moving it to the Recycle Bin.
Ctrl+N	Opens current location in a new window.
Ctrl+Shift while dragging an item	Create shortcut to selected item.

Figure 3-56: Windows Explorer shortcuts


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
420 / 867
100%
Find

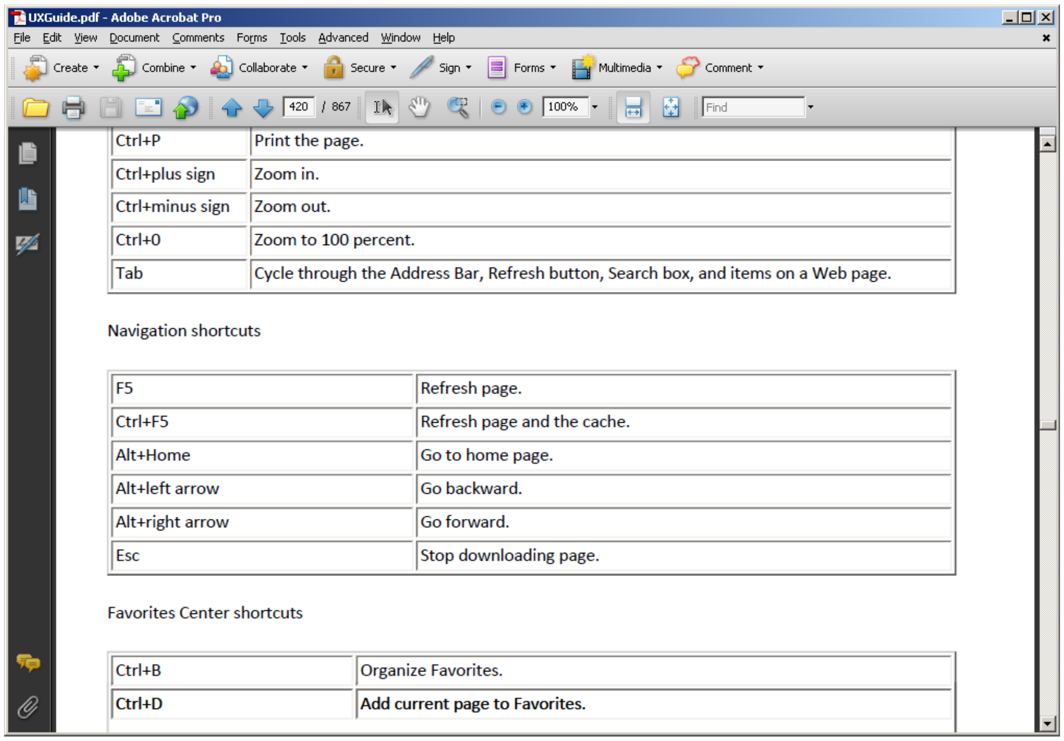
Left Alt+Shift	Switch input languages or keyboard layouts (available and configurable when the user installed multiple keyboard layouts).
Ctrl+Shift	Switch keyboard layouts or input languages (available and configurable when the user installed multiple keyboard layouts).
Ctrl or left Alt+Shift+~, number (0~9), or grave accent key	Shortcut key for input languages (available and configurable when the user installed multiple keyboard layouts).
Esc	Cancel the current task or search.

Windows Internet Explorer® shortcuts

General shortcuts

Key	Meaning
F11, Alt+Enter	Turn Full Screen Mode on or off.
Ctrl+A	Select all items on the page.
Ctrl+F	Find a word or phrase on a page.
Ctrl+L	Open the Open dialog box.
Ctrl+N	Open the current Web page in a new window.

Figure 3-57: Windows Internet Explorer shortcuts: General shortcuts



UXGuide.pdf - Adobe Acrobat Pro

Ctrl+P	Print the page.
Ctrl+plus sign	Zoom in.
Ctrl+minus sign	Zoom out.
Ctrl+0	Zoom to 100 percent.
Tab	Cycle through the Address Bar, Refresh button, Search box, and items on a Web page.

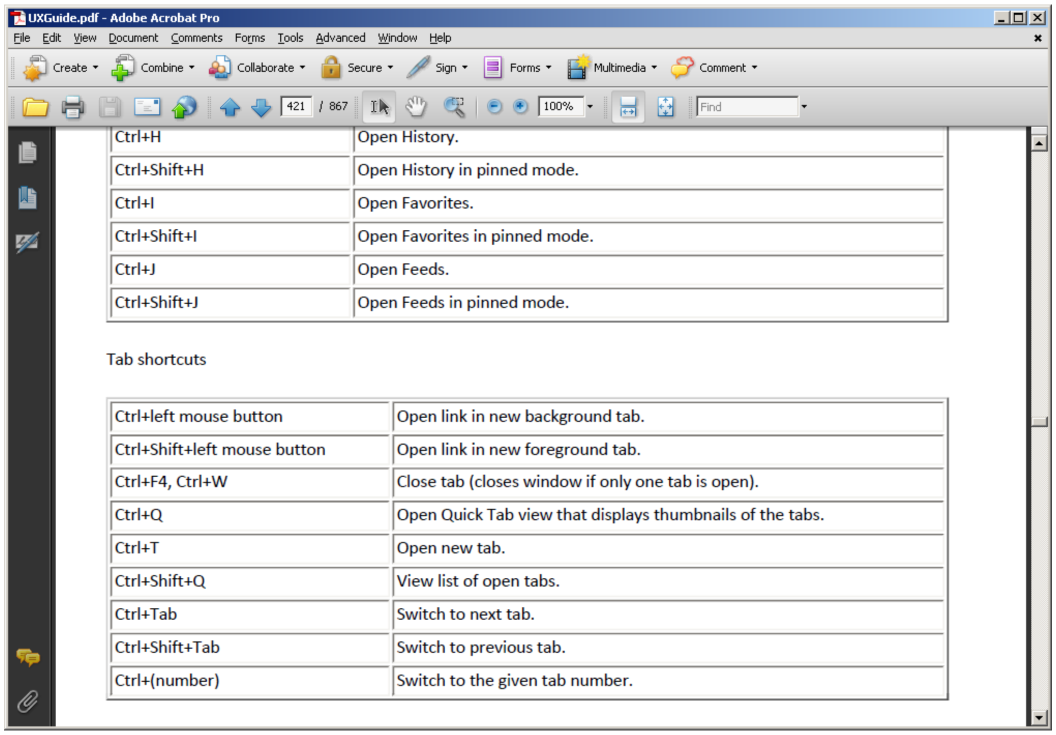
Navigation shortcuts

F5	Refresh page.
Ctrl+F5	Refresh page and the cache.
Alt+Home	Go to home page.
Alt+left arrow	Go backward.
Alt+right arrow	Go forward.
Esc	Stop downloading page.

Favorites Center shortcuts

Ctrl+B	Organize Favorites.
Ctrl+D	Add current page to Favorites.

Figure 3-58: Navigation and Favorites Center shortcuts



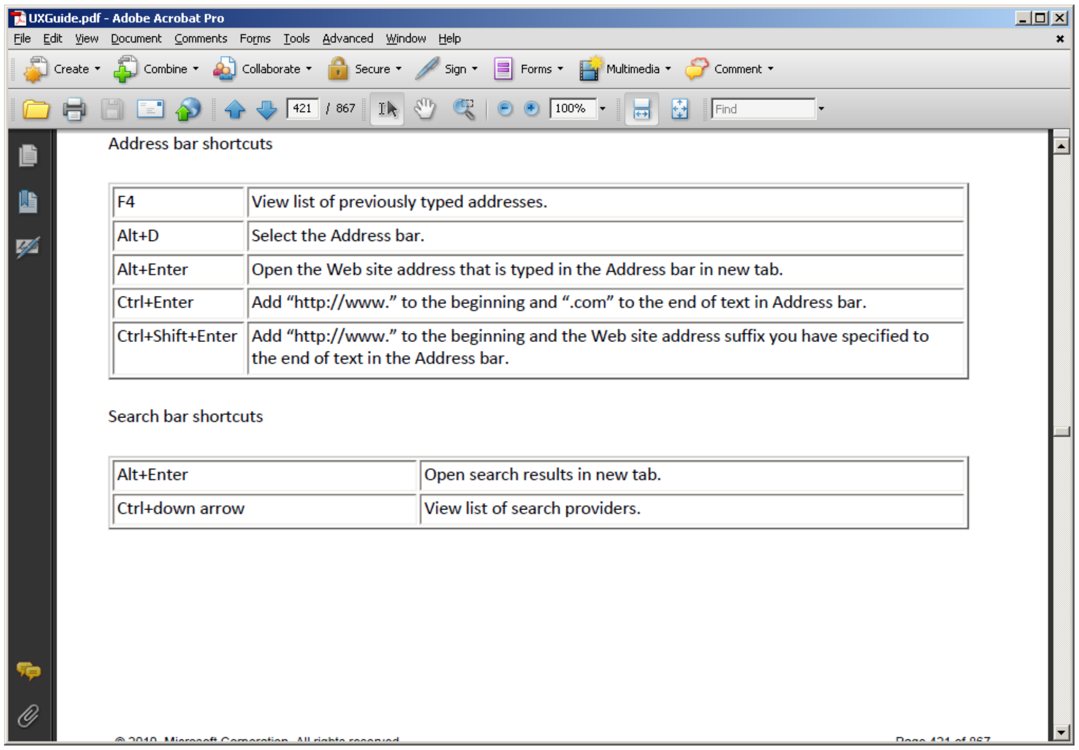
UXGuide.pdf - Adobe Acrobat Pro

Ctrl+H	Open History.
Ctrl+Shift+H	Open History in pinned mode.
Ctrl+I	Open Favorites.
Ctrl+Shift+I	Open Favorites in pinned mode.
Ctrl+J	Open Feeds.
Ctrl+Shift+J	Open Feeds in pinned mode.

Tab shortcuts

Ctrl+left mouse button	Open link in new background tab.
Ctrl+Shift+left mouse button	Open link in new foreground tab.
Ctrl+F4, Ctrl+W	Close tab (closes window if only one tab is open).
Ctrl+Q	Open Quick Tab view that displays thumbnails of the tabs.
Ctrl+T	Open new tab.
Ctrl+Shift+Q	View list of open tabs.
Ctrl+Tab	Switch to next tab.
Ctrl+Shift+Tab	Switch to previous tab.
Ctrl+(number)	Switch to the given tab number.

Figure 3-59: Tab shortcuts


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
421 / 867
100%
Find

Address bar shortcuts

F4	View list of previously typed addresses.
Alt+D	Select the Address bar.
Alt+Enter	Open the Web site address that is typed in the Address bar in new tab.
Ctrl+Enter	Add "http://www." to the beginning and ".com" to the end of text in Address bar.
Ctrl+Shift+Enter	Add "http://www." to the beginning and the Web site address suffix you have specified to the end of text in the Address bar.

Search bar shortcuts

Alt+Enter	Open search results in new tab.
Ctrl+down arrow	View list of search providers.

Figure 3-60: Address bar and Search bar shortcuts

3.40 Mouse and Pointers (Page 422)

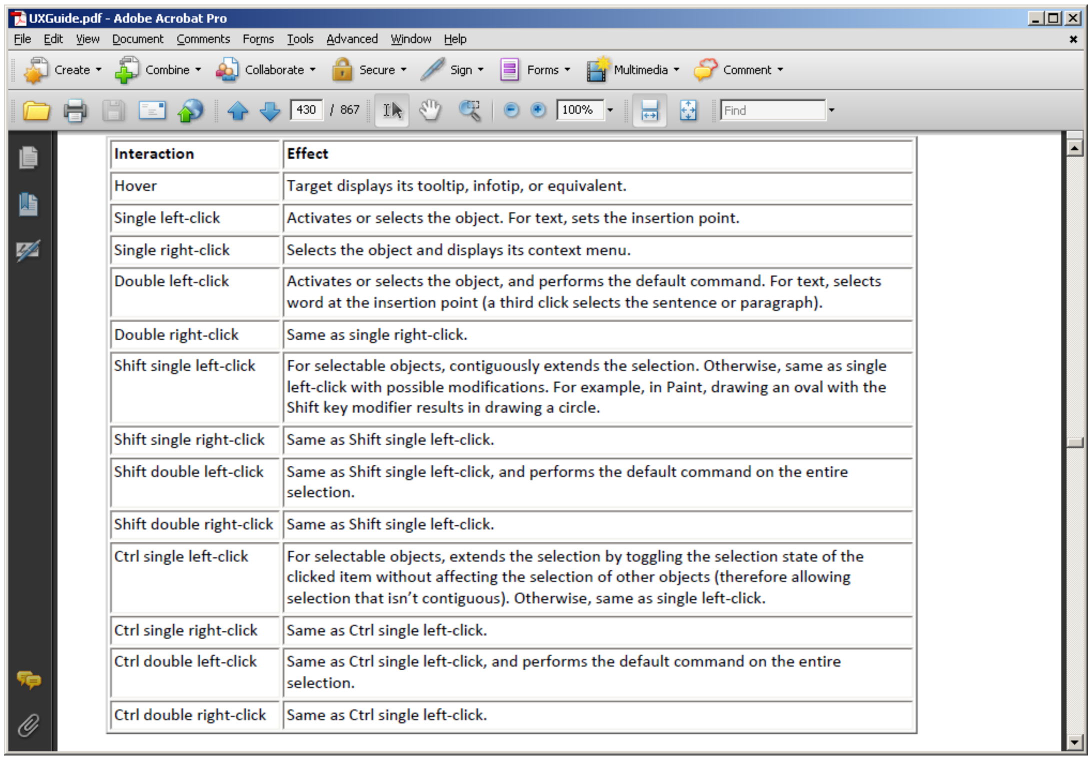
3.40.1 Guidelines

3.40.1.1 Click Affordance

- **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, 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.

3.40.1.2 Standard Mouse Button Interactions

The following table summarizes the mouse button interactions that apply in most cases.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
430 / 867
100%
Find

Interaction	Effect
Hover	Target displays its tooltip, infotip, or equivalent.
Single left-click	Activates or selects the object. For text, sets the insertion point.
Single right-click	Selects the object and displays its context menu.
Double left-click	Activates or selects the object, and performs the default command. For text, selects word at the insertion point (a third click selects the sentence or paragraph).
Double right-click	Same as single right-click.
Shift single left-click	For selectable objects, contiguously extends the selection. Otherwise, same as single left-click with possible modifications. For example, in Paint, drawing an oval with the Shift key modifier results in drawing a circle.
Shift single right-click	Same as Shift single left-click.
Shift double left-click	Same as Shift single left-click, and performs the default command on the entire selection.
Shift double right-click	Same as Shift single left-click.
Ctrl single left-click	For selectable objects, extends the selection by toggling the selection state of the clicked item without affecting the selection of other objects (therefore allowing selection that isn't contiguous). Otherwise, same as single left-click.
Ctrl single right-click	Same as Ctrl single left-click.
Ctrl double left-click	Same as Ctrl single left-click, and performs the default command on the entire selection.
Ctrl double right-click	Same as Ctrl single left-click.

Figure 3-61: Interactions and effects

3.40.1.3 Mouse Interaction

- **Make click targets at least 16x16 pixels so that they can be easily clicked by any input device.** For **touch**, the recommended minimum control size is 23 × 23 pixels (13 × 13 DLUs). Consider dynamically changing the size of small targets when the user is pointing to make them easier to acquire.
- **Make splitters at least five pixels wide so that they can be easily clicked by any input device.** Consider dynamically changing the size of small targets when the user is pointing to make them easier to acquire.
- **Provide users a margin of error spatially.** Allow for some mouse movement (for example, three pixels) when users release a mouse button. Users sometimes move the mouse slightly as they release the mouse button, so the mouse position just before button release better reflects the user's intention than the position just after.
- **Provide users a margin of error temporally.** Use the system double-click speed to distinguish between single and double clicks.

- **Have clicks take effect on mouse button up.** Allow users to abandon mouse actions by removing the mouse from valid targets before releasing the mouse button. For most mouse interactions, pressing a mouse button only indicates the selected target and releasing the button activates the action. Auto-repeat functions (such as pressing a scroll arrow to continuously scroll) are an exception.
- **Capture the mouse** for selecting, moving, resizing, splitting, and dragging.
- Use the Esc key to let users abandon compound mouse interactions such as moving, resizing, splitting, and dragging.
- **If an object doesn't support double clicks but users are likely to assume it does, interpret a "double click" as one single click.** Assume the user intended a single action instead of two.
- **Ignore redundant mouse clicks while your program is inactive.** For example, if the user clicks a button 10 times while a program is inactive, interpret that as a single click.
- **Don't use double drags or chords.** A double drag is a drag action commenced with a double-click, and a chord is when multiple mouse buttons are pressed simultaneously. These interactions aren't standard, aren't discoverable, are difficult to perform, and are most likely performed accidentally.
- **Don't use Alt as a modifier for mouse interactions.** The Alt key is reserved for toolbar access and access keys.
- **Don't use Shift-Ctrl as a modifier for mouse interactions.** Doing so would be too difficult to use.
- **Make hover redundant.** To make your program touchable, take full advantage of hover but only in ways that are not required to perform an action. This usually means that an action can also be performed by clicking, but not necessarily in exactly the same way. Hover isn't supported by most touch technologies, so users with such touchscreens can't perform any tasks that require hovering.

3.40.1.4 Mouse Wheel

- Make the mouse wheel affect the control, pane, or window that the pointer is currently over. Doing so avoids unintended results.
- Make the mouse wheel take effect without clicking or having input focus. Hovering is sufficient.
- Make the mouse wheel affect the object with the most specific scope. For example, if the pointer is over a scrollable list box control in a scrollable pane within a scrollable window, the mouse wheel affects the list box control.
- Don't change the input focus when using the mouse wheel.
- Give the mouse wheel the following effects: For scrollable windows, panes, and controls:

  - Rotating the mouse wheel scrolls the object vertically, where rotating up scrolls up. For the wheel to have natural mapping, rotating the mouse wheel should never scroll horizontally because doing so is disorienting and unexpected.
  - **If the Ctrl key is pressed, rotating the mouse wheel zooms the object,** where rotating up zooms in and rotating down zooms out.
  - **Tilting the mouse wheel scrolls the object horizontally.**
- For zoomable windows and panes (without scrollbars):
  - **Rotating the mouse wheel zooms the object,** where rotating up zooms in and rotating down zooms out.
  - Tilting the mouse wheel has no effect.
- For tabs:
  - **Rotating the mouse wheel can change the current tab,** regardless of the orientation of the tabs.
  - Tilting the mouse wheel has no effect.
- If the Shift and Alt keys are depressed, the mouse wheel has no effect.
- **Use the Windows system settings for the vertical scroll size (for rotating) and horizontal scroll size (for tilting).** These settings are configurable through the Mouse control panel item.
- **Make rotating the mouse wheel more rapidly result in scrolling more rapidly.** Doing so allows users to scroll large documents more efficiently.
- **For scrollable windows, consider having clicking the mouse wheel button put the window in "reader mode."** Reader mode plants a special scroll origin icon and scrolls the window in a direction and speed relative to the scroll origin.

3.40.1.5 Hiding The Pointer

Don't hide the pointer. Exceptions: Presentation applications running in full screen presentation mode may hide the pointer. However, the pointer must be restored immediately when users move the mouse, and can be rehidden after two seconds of inactivity.

Environments without a mouse (such as kiosks) can permanently hide the pointer.

By default, Windows hides the pointer while the user is typing in a text box. This Windows system setting is configurable through the Mouse control panel item.

3.40.1.6 Activity Pointers

The activity pointers in Windows are the busy pointer () and the working in background pointer ().

- Display the busy pointer when users have to wait more than one second for an action to complete. Note that the busy pointer has no hot spot, so users can't click anything while it is displayed.
- Display the working in background pointer when users have to wait more than one second for an action to complete, but the program is responsive and there is no other visual feedback that the action isn't complete.
- Don't combine activity pointers with progress bars or progress animations.

3.40.1.6.1 Caret

- **Don't display the caret until the text input window or control has input focus.** The caret suggests input focus to users, but a window or control can display the caret without input focus. Of course, don't steal input focus so that an out-of- context dialog box can display the caret.
- **Place the caret where users are most likely to type first.** Usually this is either the last place the user was typing or at the end of the text.

3.40.1.6.2 Accessibility

- For users who can't use the mouse at all, make the mouse redundant with the keyboard. Users should be able to do everything with the keyboard that they can with the mouse, except actions for which fine motor skills are essential, such as drawing and game playing.
- Users should be able to do everything with the mouse that they can with the keyboard, except efficient text entry.
- For users with limited ability to use the mouse: Don't make double-clicking and dragging the only way to perform an action.

3.41 Touch (Page 436)

3.41.1 Guidelines

3.41.1.1 Control Usage

- **Prefer using common controls.** Most common controls are designed to support a good touch experience.
- **Choose custom controls that are designed to support touch.** You might need to have custom controls to support your program's special experiences. Choose custom controls that:
  - Can be sized large enough to be easily touchable.
  - When manipulated, move and react the way real-world objects move and react, such as by having momentum and friction.
  - Are forgiving by allowing users to easily correct mistakes.

- Are forgiving of inaccuracy with clicking and dragging. Objects that are dropped near their destination should fall into the correct place.
- Have feedback that is clearly visible even when the finger is over the control, such a ripple effect.

- **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.
- **Provide appropriate default values.** Select the safest (to prevent loss of data or system access) and most secure option by default. If safety and security aren't factors, select the most likely or convenient option, thereby eliminating unnecessary interaction.
- **Provide text autocompletion.** Provide a list of most likely or recently input values to make text input much easier.
- **For important tasks that use multiple selection, if a standard multiple-selection list is normally used, provide an option to use a check box list instead.**

3.41.1.2 Control Sizing

- **For common controls, use the recommended control sizes.** The recommended control sizing satisfies the 23 × 23 pixel (13 × 13 DLU) minimum size, except for check boxes and radio buttons (their text width compensates somewhat), spin controls (which aren't usable with touch but are redundant), and splitters.
- **For command buttons used for the most important or frequently used commands, use a minimum size of 40 × 40 pixels (23 × 22 DLUs) whenever practical.** Doing so yields better speed and accuracy, and also feels more comfortable to users.
  - **For other controls: Use larger click targets.** For small controls, make the target size larger than the statically visible UI element. For example, 16 × 16 pixel icon buttons can have a 23 × 23 pixel click target buttons, and text elements can have selection rectangles 8 pixels wider than the text and 23 pixels high.
  - **Use redundant click targets.** It's acceptable for click targets to be smaller than the minimum size if that control has redundant functionality.
- **Respect system metrics.** Use system metrics for all sizes—don't hardwire sizes. If necessary, users can change the system metrics or dpi to accommodate their needs. However, treat this as a last resort because users shouldn't normally have to adjust system settings to make UI usable.

3.41.1.3 Control Layout And Spacing

- **Choose a layout that places controls close to where they are most likely going to be used.** Keep task interactions within a small area whenever possible. Avoid long distance hand movements, especially for common tasks and for drags.
- **Use the recommended spacing.** The recommended spacing is touch-friendly. However, if your program can benefit from larger sizing and spacing, consider the recommended sizing and spacing to be minimums when appropriate.
- **Interactive controls should either be touching or preferably have at least 5 pixels (3 DLUs) of space between them.** Doing so prevents confusion when users tap outside their intended target.
- **Consider adding more than the recommended vertical spacing within groups of controls,** such as command links, check boxes, and radio buttons, as well as between the groups. Doing so makes them easier to differentiate.
- **Consider adding more than the recommended vertical spacing dynamically when an action is initiated using touch.** Doing so makes objects easier to differentiate, but without taking more space when using a keyboard or mouse. Increase the spacing by a third of its normal size or at least 8 pixels.

3.41.1.4 Interaction

- **Make hover redundant.** Take full advantage of hover, but only in ways that are not required to perform an action. This usually means that the action can also be performed by clicking, but not necessarily in exactly the same way. Hover isn’t supported by most touch technologies, so users with such touchscreens can’t perform any tasks that require hovering.
  - For programs that need text input, fully integrate the touch keyboard feature by: Providing appropriate default values for user input.
  - Providing auto-complete suggestions when appropriate.
- **Allow users to zoom the content UI** if your program has tasks that require editing text. Consider automatically zooming to 150% when touch is used.
- **Provide smooth, responsive panning and zooming wherever appropriate.** Redraw quickly after a pan or zoom to remain responsive. Doing so is necessary to make direct manipulation feel truly direct.
- **During a pan or zoom, make sure that the contact points stay under the finger throughout the gesture.** Otherwise, the pan or zoom is difficult to control.
- **Because gestures are memorized, assign them meanings that are consistent across programs.** Don’t give different meanings to gestures with fixed semantics. Use an appropriate program-specific gesture instead.

3.41.1.5 Windows Touch Gestures

Use the following gestures whenever applicable to your program. These gestures are the most useful and natural.

- **Panning** *Entry state:* One or two fingers in contact with the screen. *Motion:* Drag, with any additional fingers remaining in same position relative to each other. *Exit state:* Last finger up ends the gesture. *Effect:* Move the underlying object directly and immediately as the fingers move. Be sure to keep the contact point under the finger throughout the gesture.
- **Zoom** *Entry state*: Two fingers in contact with the screen at the same time. *Motion:* Fingers move apart or together (pinch) along an axis. *Exit state:* Any finger up ends the gesture or the fingers break the axis. *Effect:* Zoom the underlying object in or out directly and immediately as the fingers separate or approach on the axis. Be sure to keep the contact points under the finger throughout the gesture.
- If animated carefully, allowing users to zoom while panning can be a powerful, efficient interaction.
- **Rotate** *Entry state:* Two fingers in contact with the screen at the same time. *Motion:* One or both fingers rotate around the other, moving perpendicular to the line between them. *Exit state:* Any finger up ends the gesture. *Effect:* Rotate the underlying object the same amount as the fingers have rotated. Be sure to keep the contact points under the finger throughout the gesture.
- Rotation makes sense only for certain types of objects, so it's not mapped to a system Windows interaction. Rotation is often done differently by different people. Some people prefer to rotate one finger around a pivot finger, while others prefer to rotate both fingers in a circular motion. Most people use a combination of the two, with one finger moving more than the other. While smooth rotation to any angle is the best interaction, in many contexts, such as photo viewing, it is best to settle to the nearest 90 degree rotation once the user lets go. In photo editing, a small rotation can be used to straighten the photo.
- **Two-finger tap** *Entry state:* Two fingers in contact with the screen at the same time. *Motion:* No motion. *Exit state:* Any finger up ends the gesture. *Effect:* Alternatively zooms or restores the default view for the object between the fingers.
- **Press and tap** *Entry state:* One finger in contact with the screen, followed by a second finger. *Motion:* No motion. *Exit state:* Second finger up ends the gesture. *Effect:* Performs a right click for the object under the first finger.

3.41.1.6 Forgiveness

- **Provide an Undo command.** Ideally, you should provide a simple way to undo all commands, but your program may have some commands whose effect cannot be undone.

- **Whenever practical, provide good feedback on finger down, but don't take actions until finger up.** Doing so allows users to correct mistakes before they make them.
- **Whenever practical, allow users to correct mistakes easily.** If an action takes effect on finger up, allow users to correct mistakes by sliding while the finger is still down.
- **Whenever practical, indicate that a direct manipulation can't performed by resisting the movement.** Allow the movement to happen, but have the object settle back in place when released to clearly indicate that the action was recognized but can't be done.
- **Have clear physical separation between frequently used commands and destructive commands.** Otherwise, users might touch destructive commands accidentally. A command is considered destructive if its effect is widespread and either it cannot be easily undone or the effect isn't immediately noticeable.
- **Confirm commands for risky actions or commands that have unintended consequences.** Use a confirmation dialog box for this purpose.
- **Consider confirming any other actions that users tend to do accidentally when using touch, and which either go unnoticed or are difficult to undo.** Normally, these are called **routine confirmations** and are discouraged based on the assumption that users don't often issue such commands by accident with a mouse or keyboard. To prevent unnecessary confirmations, present these confirmations only if the command was initiated using touch.

3.42 Pen (Page 453)

3.42.1 Guidelines

3.42.1.1 Control Usage

- **Prefer using common controls.** Most common controls are designed to support a good pen experience.
- **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.
- **Provide appropriate default values.** Select the safest (to prevent loss of data or system access) and most secure option by default. If safety and security aren't factors, select the most likely or convenient option, thereby eliminating unnecessary interaction.
- **Provide text autocompletion.** Provide a list of most likely or recently input values to make text input much easier.

- **For important tasks that use multiple selection, if a standard multiple-selection list is normally used, provide an option to use a check box list instead.**
- **Respect system metrics.** Use system metrics for all sizes—don't hardwire sizes. If necessary, users can change the system metrics or dpi to accommodate their needs. However, treat this as a last resort because users shouldn't normally have to adjust system settings to make UI usable.

3.42.1.2 Control Sizing, Layout, And Spacing

- **For common controls, use the recommended control sizes.** These are large enough for a good pen experience, except for spin controls (which aren't usable with a pen but are redundant).
- **Choose a layout that places controls close to where they are most likely going to be used.** Keep task interactions within a small area whenever possible. Avoid long distance hand movements, especially for common tasks and for drags.
- **Use the recommended spacing.** The recommended spacing is pen-friendly.
- **Interactive controls should either be touching or preferably have at least 5 pixels (3 DLUs) of space between them.** Doing so prevents confusion when users tap outside the intended target.
- **Consider adding more than the recommended vertical spacing within groups of controls,** such as command links, check boxes, and radio buttons, as well as between the groups. Doing so makes them easier to differentiate.

3.42.2 Interaction

- **For programs designed to accept handwriting, enable default inking.** Default inking allows users to input ink by just starting to write, without having to tap, give a command, or do anything special. Doing so enables the most natural experience with a pen. For programs not designed to accept handwriting, handle pen input in text boxes as selection.
- **Allow users to zoom the content UI** if your program has tasks that require editing text. Consider automatically zooming to 150% when a pen is used.
- **Because gestures are memorized, assign them meanings that are consistent across programs.** Don't give different meanings to gestures with fixed semantics. Use an appropriate program-specific gesture instead.

3.42.3 Handedness

If a window is contextual, always display it near the object that it was launched from. Place it out of the way so that the source object isn't covered by the window.

- If displayed using the mouse, when possible place the contextual window offset down and to the right.
- If displayed using a pen, when possible place the contextual window so as not to be covered by the user's hand. For right-handed users, display to the left; otherwise display to the right.

3.42.4 Forgiveness

- **Provide an undo command.** Ideally, you should provide undo for all commands, but your program may have some commands whose effect cannot be undone.
- **Provide good hover feedback.** Clearly indicate when the pen is over a clickable target. Such feedback is a great way to prevent accidental manipulation.
- **Whenever practical, provide good feedback on pen down, but don't take actions until a move or pen up.** Doing so allows users to correct mistakes before they make them.
- **Whenever practical, allow users to correct mistakes easily.** If an action takes effect on pen up, allow users to correct mistakes by sliding while the pen is still down.

3.43 Accessibility (Page 463)

3.43.1 Guidelines

3.43.1.1 General

- Don't disrupt or disable activated features of the operating system or other products that are identified as accessibility features. You can identify these features by referring to the documentation of the operating system or product in question.
- Don't force users to interact with your program as the top window on the screen. If a function or a window is required continuously for users to perform a task, that window should always remain visible, if the user so chooses, regardless of its position relative to other windows. For example, if the user has a movable on-screen keyboard that is on top of all other windows so that it is visible at all times, your program should never obscure it by mandatory placement at the top of the Z order.
- Use system colors, fonts, and common controls whenever possible. By doing so, you significantly reduce the number of accessibility issues users encounter.

3.43.1.2 Addressing Particular Impairments

3.43.2 Visual

- **Never rely on color alone to convey meaning.** Use color only as a means of reinforcing the meaning provided by text, design, location, or sound.
- **Use alternative (alt) text infotips to describe graphics.**
- **Don't use text in graphics.** Users with visual impairments may have graphics turned off (for example, in a Web browser), or may simply not see or look for text placed in graphics.
- **Ensure that dialog boxes and windows have meaningful names,** so that a user who is hearing rather than seeing the screen (for example, using a screen reader) gets appropriate contextual information.
- **Respect the user's settings for visual display** by always obtaining font typefaces, sizes, and colors, Windows display element sizes, and system configuration settings from the Theme and GetSystemMetrics APIs.
- **Keep balloon text concise** so that it is easier to read and minimizes disruption to screen readers.

3.43.3 Hearing

- **Never rely on sound alone to convey meaning.** Use sound only as a means of reinforcing the meaning provided by text, design, location, or color.
- **Enable users to control the volume of audio output.** Use the Windows Volume Mixer for this purpose. For more information, see **Sound**.
- **Target your program's sound to occur in a range between 500 Hz and 3000 Hz** or be easily adjustable by the user into that range. Sounds in this range are most likely to be detectable by people with hearing impairment.

3.43.4 Dexterity

- Make UI timeout values relative to GetDoubleClickTime() instead of using absolute times. Doing so adjusts the timeouts to the speed of the user.
- Assign access keys to all menu items so that users who prefer working with the keyboard have the same ability to navigate your program as users who work with the mouse.
- Don't make double-clicking and dragging the only way to perform an action. These can be difficult movements for some users.
- Don't remove menu bars from your program. Menu bars are easier than toolbars for keyboard users to access. If you don't want the menu bar visible by default, hide it instead.

- Make Help accessible from the keyboard by providing tab stops for Help buttons and links.
- To improve awareness of the access key assignments in your program, you can display them at all times. In Control Panel, go to the Ease of Access Center, and click Make the keyboard easier to use; then select the Underline keyboard shortcuts and access keys check box.

3.43.5 Cognitive

- Use progressive disclosure to hide complexity.
- Use icons, toolbars, and other visual aids to reduce cognitive load of reading text.
- When possible, provide auto-complete functionality in text boxes and editable drop-down lists, so that users don't have to type the entire name of commands, file names, or similar choices from a limited set of options. This reduces cognitive load for all users, and reduces the amount of typing for users for whom spelling or typing is difficult, slow, or painful.
- Demonstrate difficult concepts in Help by including tutorials and animations. Note that animations can be difficult for users with seizure impairment, and therefore should be used only when necessary.

3.43.6 Seizure

- Don't use flashing or blinking text, objects, or other elements having a flash or blink frequency in the range between 2–55 Hz.
- Limit use of animations. Some users are particularly sensitive to screen movement, especially in the periphery of their visual field. If you use animation to draw attention to something, make sure that attention is deserved and worthy of interrupting the user.

3.43.7 Speech Or Language

- **Organize and write clear, concise, easily understood text.** Usability tests show that unfolding key information at the end of a phrase improves comprehension.
- **Access keys**
- **Prefer characters with wide widths,** such as w, m, and capital letters.
- **Prefer a distinctive consonant or a vowel,** such as “x” in “Exit.”
- Avoid using characters that make the underline difficult to see, such as (from most problematic to least problematic):
  - Characters that are only one pixel wide, such as “i” and “l”.
  - Characters with descenders, such as “g”, “j”, “p”, “q”, and “y”.
  - Characters next to a letter with a descender.

3.43.8 Menu Access Keys

- Assign access keys to all menu items. No exceptions.
- For dynamic menu items (such as recently used files), assign access keys numerically.
- Assign unique access keys within a menu level. You can reuse access keys across different menu levels.
- **Make access keys easy to find:**
  - For the most frequently used menu items, choose characters at the beginning of the first or second word of the label, preferably the first character.
  - For less frequently used menu items, choose letters that are a distinctive consonant or a vowel in the label.

3.43.9 Dialog Box Access Keys

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.

- **Group labels.** Normally, the individual controls within a group are assigned access keys, so the group label doesn’t need one. However, assign an access key to the group label and not the individual controls if there is a shortage of access keys.
- **Generic Help buttons,** which are accessed with F1.
- **Link labels.** There are often too many links to assign unique access keys, and link underscores hide the access key underscores. Have users access links with the Tab key instead.
- **Tab names.** Tabs are cycled using Ctrl+Tab and Ctrl+Shift+Tab.
- **Browse buttons labeled “...”.** These can’t be assigned access keys uniquely.
- **Unlabeled controls,** such as spin controls, graphic command buttons, and unlabeled progressive disclosure controls.
- **Nonlabel static text or labels for controls that aren’t interactive,** such as progress bars.
- **Assign commit button access keys first to ensure that they have the standard key assignments.** If there isn’t a standard key assignment, use the first letter of the first word. For example, the access key for Yes and No commit buttons should always be “Y” and “N”, regardless of the other controls in the dialog box.

- **For negative commit buttons (other than Cancel) phrased as a "Don't", assign the access key to the "n" in "Don't".** If not phrased as a "Don't", use the standard access key assignment or assign the first letter of the first word. By doing so, all Don'ts and No's have a consistent access key.
- **To make access keys easy to find, assign the access keys to a character that appears early in the label,** ideally the first character, even if there is a keyword that appears later in the label.

3.43.10 Text

- **Use colons at the end of external control labels.** Some assistive technologies look for colons to identify control labels.
- **Position labels consistently relative to the elements that they are labeling.** This helps assistive technology correctly associate the labels with their corresponding controls, and helps users of screen enlargers know where to look for a label or control.
- **Don't use text to draw lines, boxes or other graphical symbols.** Characters used in this way can confuse users of screen readers. For example, a box drawn with the letter "X" around an area of text is read by screen-reader software as "X X X X X X" on the first line, followed by "X" and the content and "X".

3.44 Window Management (Page 475)

3.44.1 Guidelines

3.44.1.1 General

- Support the minimum Windows effective resolution of 800 × 600 pixels. For critical user interfaces (UIs) that must work in safe mode, support an effective resolution of 640 × 480 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 1024 × 768 pixels. Automatically resize these windows for lower screen resolutions in a way that is still functional.
- Be sure to test your windows in 96 dpi (100%) at 800 × 600 pixels, 120 dpi (125%) at 1024 × 768 pixels, and 144 dpi (150%) at 1200 × 900 pixels. 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.

- Resizable windows no longer must show the resize glyph in the lower-right corner, because: All sides and edges of a window are resizable, not just the lower-right corner.
- The glyph requires a status bar to display, yet many resizable windows don't provide status bars.
- The resizable window borders and resize pointers are more effective at communicating that a window is resizable than the resize glyph.

3.44.1.2 Title Bar Controls

Use the title bar controls as follows:

- **Close**. All primary and secondary windows with a standard window frame should have a Close button on the title bar. Clicking Close has the effect of canceling or closing the window.
- **Minimize.** All primary windows and long-running modeless secondary windows (such as progress dialogs) should have a Minimize button. Clicking Minimize reduces the window to its taskbar button. Consequently, windows that can be minimized require a title bar icon.
- **Maximize/Restore down.** All resizable windows should have a Maximize/Restore down button. Clicking Maximize displays the window in its largest size, which for most windows is full screen; whereas clicking Restore down displays the window in its previous size. However, some windows don't benefit from using a full screen, so these windows should maximize to their largest useful size.

3.44.1.3 Window Size

- **Choose a default window size appropriate for its contents.** Don't be afraid to use larger initial window sizes if you can use the space effectively.
- **Use resizable windows whenever practical to avoid scroll bars and truncated data.** Windows with dynamic content and lists benefit the most from resizable windows.
- **For text documents, consider a maximum line length of 65 characters** to make the text easy to read. (Characters include letters, punctuation, and spaces.)
- Fixed-sized windows: **Must be entirely visible and sized to fit within the work area.**
- Resizable windows: **May be optimized for higher resolutions, but sized down as needed at display time to the actual screen resolution.**
- **For progressively larger window sizes, must show progressively more information.** Make sure that at least one window portion or control has resizable content.

- **Should avoid default restored sizes that are maximized or near maximized.** Instead, choose a default size that is typically the most useful without being full screen. Assume that users will maximize the window instead of resizing to make it full screen.
- **Should 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.
- **Should change the presentation if doing so makes the content usable at smaller sizes.**

3.44.1.4 Window Location

- For the following guidelines, "centering" means to bias vertical placement slightly towards the top of the monitor, instead of placing exactly in the middle. Put 45% of the space between the top of the monitor/owner and the window top, and 55% between the bottom of the monitor/owner and the window bottom. Do this because the eye is naturally biased towards the top of the screen.
- If a window is contextual, always display it near the object that it was launched from. Place it out of the way so that the source object isn't covered by the window. If displayed using the mouse, when possible place it offset down and to the right.

Show contextual windows near the object that it was launched from.

If displayed using a pen, when possible place it so as not to be covered by the user's hand. For right-handed users, display to the left; otherwise display to the right.

- Place progress dialogs out of the way in the lower-right corner of the active monitor.
  - **If a window isn't related to the current context or user action, place it away from the current pointer location.** Doing so prevents accidental interaction.
  - **If a window is a top-level application or document, always cascade its origin off the upper-left corner of the monitor.** If created by the active program, use the active monitor; otherwise, use the default monitor.
  - **If a window is a top-level utility, *always* display it "centered" in the monitor.** If created by the active program, use the active monitor; otherwise, use the default monitor.
  - **If a window is an owned window, *initially* display it "centered" on top of the owner window.** For subsequent display, consider displaying it in its last location (relative to the owner window) if doing so is likely to be more convenient.

- **For modeless dialogs, always display initially on top of the owner window to make them easy to find.** However, if the user activates the owner window, that may obscure the modeless dialog.
- **If necessary, adjust the initial location so that the entire window is visible within the target monitor.** If a resizable window is larger than the target monitor, reduce it to fit.

3.44.2 Window Order (Z Order)

- **Always place owned windows on top of their owner window.** Never place owned windows under their owner windows, because most likely users won't see them.
- **Respect users' Z order selection. When users select a window, bring only the windows associated with that instance of the program (the window plus any owner or owned windows) to top of the Z order.** Don't change the order of any other windows, such as independent instances of same program.

3.44.3 Window Activation

- Respect users' window state selection. If an existing window needs attention, flash the taskbar button three times to draw attention and leave it highlighted, but don't do anything else. Don't restore or activate the window. Don't use any sound effects. Instead, let users activate the window when they are ready. Exception: If the window doesn't appear on the taskbar, bring it to the top of all the other windows and flash its title bar instead.
- Restoring a primary window should also restore all its secondary windows, even if those secondary windows have their own taskbar button. When restoring, place secondary windows on top of the primary window.

3.44.4 Input Focus

- **Windows displayed by user-initiated actions should take input focus, but only if the window is rendered immediately** (within 5 seconds). Once the window is rendered, it can take input focus once. If a window renders slowly (more than 5 seconds), users are likely to perform another task while they wait. Taking focus at this point would be an annoyance, especially if done more than once.
- **Windows that aren't immediately displayed or displayed by a system-initiated action shouldn't take input focus.** Instead, display on top without focus, and let users activate them when they are ready. **Exception:** Credential Manager.

3.44.5 Persistence

- **When a window is redisplayed, consider displaying it in the same state as last accessed.** When closing, save the monitor used, window size, location, and state (maximized vs. restore). When redisplaying, restore the saved window size, location, and state using the appropriate monitor. Also, consider making these attributes persist across program instances on a per-user basis. **Exceptions:** Don't save or make these attributes persist for windows when their usage is such that users are far more likely to want to start completely over.
- For programs likely to be used on Windows Tablet and Touch Technology computers, save two windows states for landscape and portrait modes. For more information, see **Designing for Varying Display Sizes**.
- **If the current monitor configuration prevents displaying a window using its last state:** Try to display the window using its last monitor.
- If the window is larger than the monitor, resize the window as necessary.
- Move the location toward the upper-left corner to fit within the monitor as necessary.
- If the above steps don't solve the problem, revert to the default window placement guidelines. Consider restoring the previous size, if possible.

3.45 Dialog Boxes (Page 483)

3.45.1 Guidelines

3.45.1.1 General

- Don't use scrollable dialog boxes. Don't use dialog boxes that require the use of a scroll bar to be viewed completely during normal usage. Redesign the dialog box instead. Consider using progressive disclosure or tabs.
- Don't have a menu bar or status bar. Instead, provide access to commands and status directly on the dialog box itself, or by using context menus on the relevant controls. Exception: Menu bars are acceptable when a dialog box is used to implement a primary window (such as a utility).
  – If a dialog box requires immediate attention and the program isn't active, **flash its taskbar button three times to draw attention, and leave it highlighted.** Don't do anything else: don't restore or activate the window and don't play any sound effects. Instead, respect the user's window state selection and let the user activate the window when ready.

3.45.2 Modal Dialog Boxes

- Use for critical or infrequent, one-off tasks that require completion before continuing.
- Use a delayed commit model so that changes don't take effect until explicitly committed.
- Implement using a task dialog whenever appropriate to achieve a consistent look. Task dialogs do require Windows Vista or later, so they aren't suitable for earlier versions of Windows.

3.45.3 Modeless Dialog Boxes

- Use for frequent, repetitive, on-going tasks.
- Use an immediate commit model so that changes take effect immediately.
- For modeless dialogs, use an explicit Close command button in the dialog to close the window. For both, use a Close button on the title bar to close the window.
- Consider making modeless dialog boxes dockable. Dockable modeless dialogs allow for more flexible placement.

3.45.4 Multiple Dialog Boxes

- Don't display more than one owned choice dialog at a time from an owner choice dialog. Displaying more than one makes the meaning of the commit buttons difficult for users to understand. You may display other types of dialog boxes (such question dialogs) as needed.
- For a sequence of related dialogs, consider using a multi-page dialog if possible. Use individual dialogs if they aren't clearly related.

3.45.5 Multipage Dialog Boxes

- Use a multipage dialog box instead of individual dialog boxes when you have the following sequence of *related* pages:
  - A single input page (optional)
  - A progress page
  - A single results page

    The input page is optional because the task may have been initiated somewhere else. Doing so gives the resulting experience a stable, simple, lightweight feel.
- Don't use a multipage dialog if the input page is a standard dialog. In this case the consistency of using a standard dialog is more important.

- Don't use Next or Back buttons and don't have more than three pages. Multi-page dialog boxes are for single-step tasks with feedback. They aren't wizards, which are used for multi-step tasks. Wizards have a heavy, indirect feel compared to multi-page dialog boxes.
- On the input page, use specific command buttons or command links to initiate the task.
- Use a Cancel button on the input and progress pages, and a Close button on the results page.

3.45.6 Presentation

- To make dialog boxes easy to find and access, clearly associate the dialog with its source, and work well with multiple monitors:
- Initially display dialogs "centered" on top of the owner window. 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 dialog is contextual, display it near the object from which it was launched. However, place it out of the way (preferably offset down and to the right) so that the object isn't covered by the dialog.
- For modeless dialogs, display initially on top of the owner window to make it easy to find. If the user activates the owner window, that may obscure the modeless dialog.
- If necessary, adjust the initial location so that the entire dialog is visible within the target monitor. If a resizable window is larger than the target monitor, reduce it to fit.
- When a dialog is redisplayed, consider displaying it in the same state as last accessed. On close, save the monitor used, window size, location, and state (maximized vs. restore). On redisplay, restore the saved dialog size, location, and state using the appropriate monitor. Also, consider making these attributes persist across program instances on a per-user basis.
- For resizable windows, set a minimum window size if there is a size below which the content is no longer usable. Consider altering the presentation to make the content usable at smaller sizes.
- **Don't use the Always on Top attribute. Exception:** Use only when a dialog box implements an essentially modal operation, but it needs to be suspended briefly to access the owner window. For example, when spell-checking a document, users may occasionally leave the spell check dialog box and access the document to correct errors.

3.45.7 Title Bars

- **Dialog boxes don't have title bar icons.** Title bar icons are used as a visual distinction between **primary windows** and **secondary windows**. **Exception:** If a dialog box is used to implement a primary window (such as a utility) and therefore appears on the taskbar, it does have a title bar icon. In this case, optimize the title for display on the taskbar by concisely placing the distinguishing information first.
- **Dialog boxes always have a Close button.** Modeless dialogs can also have a Minimize button. Resizable dialogs can have a Maximize button.
- **Don't disable the Close button.** Having a Close button helps users stay in control by allowing them to close windows they don't want. **Exception:** For progress dialogs, you may disable the Close button if the task must run to completion to achieve a valid state or prevent data loss.
- **The Close button on the title bar should have the same effect as the Cancel or Close button** within the dialog box. Never give it the same effect as OK.
- If the title bar caption and icon are already displayed in a prominent way near the top of the window, you can hide the title bar caption and icon to avoid redundancy. However, you still have to set a suitable title internally for use by Windows.

3.45.8 Interaction

- **When displayed, user initiated dialog boxes should always take input focus.** Program initiated dialog boxes shouldn't take input focus because the user may be interacting with another window. Such interaction misdirected at the dialog box may have unintended consequences.
- **Assign initial input focus to the control that users are most likely to interact with first**, which is usually (but not always) the first interactive control. Avoid assigning initial input focus to a Help link.
- **For keyboard navigation, tab order should flow in a logical order, generally from left to right, top to bottom.** Usually tab order follows reading order, but consider making these exceptions: Put the most commonly used controls earlier in tab order.
- Put Help links at the bottom of a dialog box, after the commit buttons in tab order.
- When assigning order, assume that users display dialog boxes for their intended purpose; so, for example, users display choice dialogs to make choices, not to review and click Cancel.
- **Pressing the Esc key always closes an active dialog box.** This is true for dialog boxes with Cancel or Close, and even if Cancel has been renamed to Close because the results can no longer be undone.

3.45.9 Access Keys

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.

- **Group labels.** Normally, the individual controls within a group are assigned access keys, so the group label doesn't need one. However, if there is a shortage of access keys, assign an access key to the group label and not the individual controls.
- **Generic Help buttons,** which are accessed with F1.
- **Link labels.** There are often too many links to assign unique access keys, and the underscores often used to signify links hide the access key underscores. Access links with the Tab key instead.
- **Tab names.** Tabs are cycled using Ctrl-Tab and Ctrl-Shift-Tab.
- **Browse buttons labeled "...".** These Browse buttons can't be assigned access keys uniquely.
- **Unlabeled controls,** such as spin controls, graphic command buttons, and unlabeled progressive disclosure controls.
- **Nonlabel static text or labels for controls that aren't interactive,** such as progress bars.
- **Whenever possible, assign access keys for commonly used commands according to the Standard Access Key Assignments.** While consistent access key assignments aren't always possible, they are certainly preferred—especially for frequently used dialog boxes.
- **Assign commit button access keys first to ensure that they have the standard key assignments.** If there isn't a standard key assignment, use the first letter of the first word. For example, the access key for Yes and No commit buttons should always be "Y" and "N", regardless of the other controls in the dialog box.
- **To make access keys easy to find, assign the access keys to a character that appears early in the label,** ideally the first character, even if there is a keyword that appears later in the label.
- **Prefer characters with wide widths,** such as w, m, and capital letters.
- **Prefer a distinctive consonant or a vowel,** such as "x" in Exit.
- **Avoid using characters that make the underline difficult to see,** such as (from most problematic to least problematic): Letters that are only one pixel wide, such as i and l.

  - Letters with descenders, such as g, j, p, q, and y.
  - Letters next to a letter with a descender.

3.45.10 Progress Dialogs

- For long-running tasks, assume that users will do something else while the task is completing. Design the task to run unattended.
- Present users with progress feedback dialog box if an operation takes longer than five seconds to complete, along with a command to cancel or stop the operation. Exception: For wizards and task flows, use a modal dialog for progress only if the task stays on the same page (as opposed to advancing to another page) and users can't do anything while waiting. Otherwise, use a progress page or in-place progress.
- If the operation is a long-running task (over 30 seconds) and can be performed in the background, use a modeless progress dialog so that users can continue to use your program while waiting.
- Modeless progress dialogs:
  - Have a Minimize button on the title bar.
  - Are displayed on the taskbar.
  - Implement modeless progress dialogs so that they continue to run even if the owner window is closed.
- Provide a command button to halt the operation if it takes more than a few seconds to complete, or has the potential never to complete. Label the button Cancel if canceling returns the environment to its previous state (leaving no side effects); otherwise, label the button Stop to indicate that it leaves the partially completed operation intact. You can change the button label from Cancel to Stop in the middle of the operation, if at some point it isn't possible to return the environment to its previous state.
- Provide a command button to pause the operation if it takes more than several minutes to complete, and it impairs users' ability to get work done. Doing so doesn't force the user to choose between completing the task and getting their work done.
- Gather as much information as you can before starting the task.
- If recoverable problems are detected, have users deal with all problems found at the end of the task. If that isn't practical, have users deal with problems as they happen.
- Don't abandon tasks as the result of recoverable errors.
- Indicate problems by turning the progress bar red.

- **If the results are clearly apparent to users, close the progress dialog automatically on successful completion.** Otherwise, use feedback only to report problems: To display simple feedback, display the feedback in the progress dialog, and change the Cancel button to Close.
- To display detailed feedback, close the progress dialog box and display an **informational dialog**.
- **Don’t use a notification for completion feedback.** Use either a progress dialog or an **action success notification**, but not both.

3.45.11 Time Remaining

- **Use the following time formats.** Start with the first of the following formats where the largest time unit isn’t zero, then change to the next format once the largest time unit becomes zero.
- **For progress bars:**
  - **If related information is shown in a colon format:** Time remaining: h hours, m minutes Time remaining: m minutes, s seconds Time remaining: s seconds**.**
  - **If screen space is at a premium:** h hrs, m mins remaining m mins, s secs remaining s seconds remaining. **Otherwise:** h hours, m minutes remaining m minutes, s seconds remaining s seconds remaining.
- **For title bars:** hh:mm remaining mm:ss remaining 0:ss remainingThis compact format shows the most important information first so that it isn’t truncated on the taskbar.
- **Make estimates accurate, but don’t give false precision.** If largest unit is hours, give minutes (if meaningful) but not seconds. **Incorrect:** hh hours, mm minutes, ss seconds
- **Keep the estimate up-to-date.** Update time remaining estimates at least every 5 seconds.
- **Focus on the time remaining** because that is the information users care about most. Give total elapsed time only when there are scenarios where elapsed time is helpful (such as when the task is likely to be repeated). If the time remaining estimate is associated with a progress bar, don’t have percent complete text because that information is conveyed by the progress bar itself.
- **Be grammatically correct.** Use singular units when the number is one. **Incorrect:** 1 minutes, 1 seconds
- **Use sentence-style capitalization.**

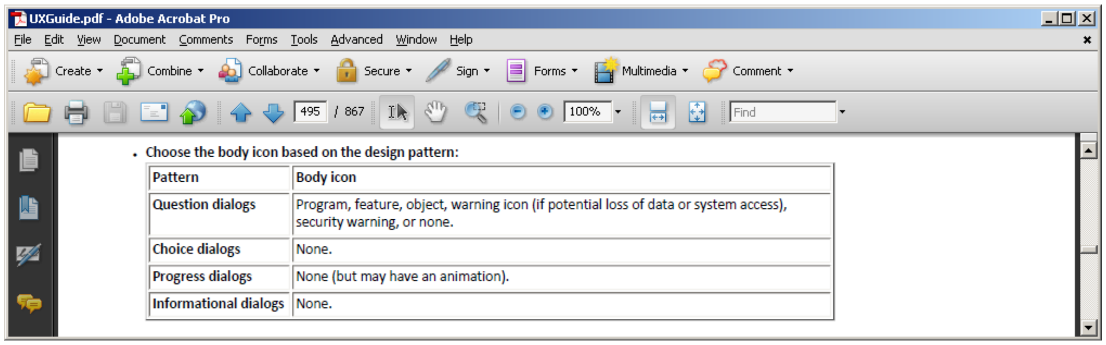
3.45.12 Icons And Graphics

3.45.12.1 Graphics

Don’t use large graphics that serve no purpose beyond filling space with eye candy. Keep the appearance simple instead.

3.45.12.2 Title Bar Icons

Dialog boxes don’t have title bar icons. Exception: If a dialog box is used to implement a primary window (such as a utility) and therefore appears on the taskbar, it does have a title bar icon.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
495 / 867
100%
Find

- Choose the body icon based on the design pattern:

Pattern	Body icon
Question dialogs	Program, feature, object, warning icon (if potential loss of data or system access), security warning, or none.
Choice dialogs	None.
Progress dialogs	None (but may have an animation).
Informational dialogs	None.

Figure 3-62: Choosing the body icon

- **Consider using icons to help users visually recognize your program’s features.** This technique is most effective when the icons are easily recognizable and used in several locations within your program.
- **Use icons to help users recognize the object in question.**
- **Consider using icons to help make features self-explanatory.**
- **Use an icon in About Box dialogs for application branding.**

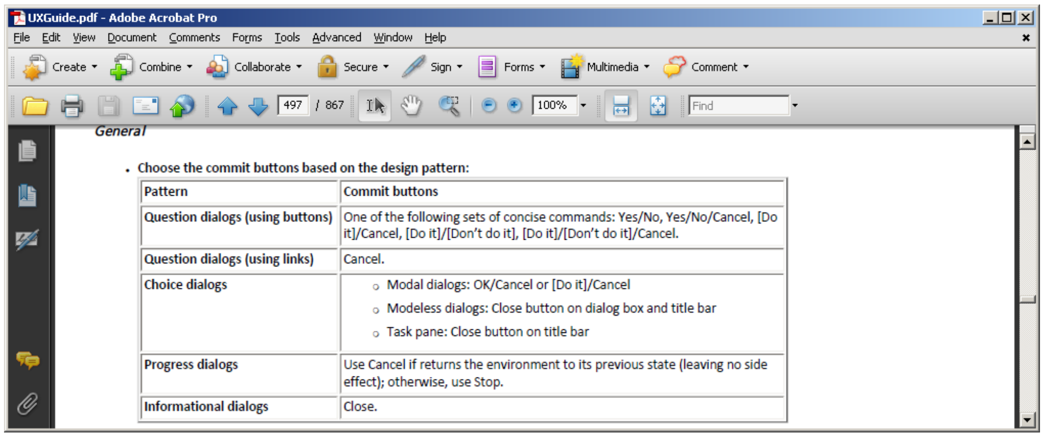
3.45.12.3 Footnote Icons

- If you have a footnote, consider using a footnote icon to summarize the footnote’s subject.
- Don’t use a footnote icon that repeats the body icon.
- Don’t use the error or information standard icons. Error conditions must be conveyed through the body icon and footnotes are always for information, making the information icon redundant. However, you can use the standard warning icon and the yellow security shield to alert users of risky consequences.

3.45.12.4 Commit Buttons

Notes:

- These guidelines don't apply to **question dialogs using command links**, because that pattern uses command links instead of buttons.
- [Do it] and [Don't do it] are affirmative and negative responses, respectively, to the main instruction.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
497 / 867
100%
Find

General

- Choose the commit buttons based on the design pattern:

Pattern	Commit buttons
Question dialogs (using buttons)	One of the following sets of concise commands: Yes/No, Yes/No/Cancel, [Do it]/Cancel, [Do it]/[Don't do it], [Do it]/[Don't do it]/Cancel.
Question dialogs (using links)	Cancel.
Choice dialogs	○ Modal dialogs: OK/Cancel or [Do it]/Cancel ○ Modeless dialogs: Close button on dialog box and title bar ○ Task pane: Close button on title bar
Progress dialogs	Use Cancel if returns the environment to its previous state (leaving no side effect); otherwise, use Stop.
Informational dialogs	Close.

Figure 3-63: Choosing the commit buttons based on the design pattern

- All commit buttons except Apply result in closing the dialog box window.
- Don't confirm commit buttons. Doing so unnecessarily can be very annoying. Exceptions: The action is potentially catastrophic.
- The action is clearly inconsistent with other actions.
- If incorrect, the action may result in a significant loss of data, time, or effort on behalf of the user. For more guidelines and examples, see Confirmations.
- Don't disable commit buttons. Exceptions: If users must elevate to make a change, disable the positive commit buttons until the user makes a change. Doing so prevents users from elevating just to close a window by forcing them to click Cancel. For more exceptions, see Disabling or removing controls vs. giving error messages.
- Right-align commit buttons in a single row across the bottom of the dialog box, but above the footnote area. Do this even if there is a single commit button (such as OK).
- Present the commit buttons in the following order:
  - OK/[Do it]/Yes
  - [Don't do it]/No
  - Cancel
  - Apply (if present)
  - Help (if present)
- If you have many related commit buttons, consolidate them using split buttons.

- Have a clear separation from commit buttons (which close the window) and all other command buttons (such as Advanced).

3.45.13 Responding To Main Instructions

- Use positive commit buttons that are specific responses to the main instruction, instead of generic labels such as OK or Yes/No. Users should be able to understand the options by reading the button text alone. Exceptions: Use Close for dialogs that don't have settings, such as informational dialogs. Never use Close for dialogs that have settings.
- Use OK when the "specific" responses are still generic, such as Save, Select, or Choose. Use OK when changing a specific setting or a collection of settings.
- For legacy dialog boxes without a main instruction, you can use generic labels such as OK. Often such dialog boxes aren't designed to perform a specific task, preventing more specific responses.
- Certain tasks require more thought and careful reading for users to make informed decisions. This is usually the case with confirmations. In such cases, you can purposely use generic commit button labels to force users to read the main instructions and prevent hasty decisions.
- Alternatively, you can add the word "anyway" to the positive commit button label to indicate that the dialog box presents a reason not to proceed and that users should read the dialog carefully before proceeding.
- Use Cancel or Close for negative commit buttons instead of specific responses to the main instruction. Quite often users realize that they don't want to perform a task once they see a dialog box. If Cancel or Close were relabeled to specific responses, users would have to carefully read all the commit buttons to determine how to cancel. Labeling Cancel and Close consistently makes them easy to find. Exceptions: Don't use Yes/Cancel. Always use Yes/No as a pair.
- Use a specific response when Cancel is ambiguous.
- Don't map generic labels to their specific meaning with text in the content area. Instead, use specific commit button labels, or a question dialog using links if the labels are lengthy.

3.45.14 Yes And No Buttons

- **Prefer specific responses to Yes and No buttons.** While there's nothing wrong with using Yes and No, specific responses can be understood more quickly, resulting in efficient decision making. However, **confirmations** usually have Yes and No buttons to make users give the confirmation **some thought** before responding.

- **Use Yes and No buttons only to respond to yes or no questions.** The main instruction should be naturally expressed as a yes or no question. Never use OK and Cancel for yes or no questions.
- **Consider phrasing the main instruction as a yes or no question if commit buttons with specific phrasing turn out to be long or awkward.** Alternatively, you can use command links for longer responses (five words or more) to the main instruction.
- **Don’t use Yes and No buttons if the meaning of the No response is unclear.** If so, use specific responses instead.

3.45.15 OK Buttons

- In modal dialogs, clicking OK means apply the values, perform the task, and close the window.
- Don’t use OK buttons to respond to questions.
- Don’t assign access keys to OK, because Enter is the access key for the default button. Doing so makes the other access keys easier to assign.
- Label OK buttons correctly. The OK button should be labeled OK, not Ok or Okay.
- Don’t use OK buttons for errors or warnings. Problems are never OK. Use Close instead.
- Don’t use OK buttons in modeless dialog boxes. Rather, modeless dialogs should use task-specific commit buttons (for example, Find). However, some modeless dialog boxes require only a Close button.

3.45.16 Cancel Buttons

- Clicking Cancel means abandon all changes, cancel the task, close the window, and return the environment to its previous state, leaving no side effect. For nested choice dialog boxes, clicking Cancel in the owner choice dialog means any changes made by owned choice dialogs are also abandoned.
- Provide a Cancel button to let users explicitly abandon changes. Dialog boxes need a clear exit point. Don’t depend on users finding the Close button on the title bar. Exception: Don’t provide a Cancel button for dialog boxes without settings. The OK and Close buttons have the same effect as Cancel in this case.
- Don’t assign access keys to Cancel, because Esc is the access key. Doing so makes the other access keys easier to assign.
- Don’t use Cancel buttons in modeless dialog boxes. Rather, use Close instead.

- Don’t disable the Cancel button. Users should always be able to cancel dialog boxes. Exception: You may disable the Cancel button in a progress dialog if there is a period during which the operation can’t be cancelled. However, a better solution is to design such operations to always be cancelable.

3.45.17 Close Buttons

- Use Close buttons for modeless dialog boxes, as well as modal dialogs that cannot be cancelled.
- Clicking Close means close the dialog box window, leaving any existing side effects. Don’t use Done, because it isn’t an imperative construction. For nested choice dialog boxes, clicking Close in the owner choice dialog means any changes made by owned choice dialogs are preserved.
- Put an explicit Close button in the dialog box body. Dialog boxes need a clear exit point. Don’t depend on users finding the Close button on the title bar.
- Make sure the Close button on the title bar has the same effect as Cancel or Close.
- Don’t assign access keys to Close, because Esc is its the access key. Doing so makes the other access keys easier to assign.

3.45.18 Apply Buttons

Don’t use Apply buttons in dialog boxes that aren’t property sheets or control panels. 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 sheet and control panels have this need.

3.45.19 Commit Buttons For Indirect Dialog Boxes

Note: Indirect dialog boxes are displayed out of context, either as an indirect result of a task or the result of a problem with a system or background process. For indirect dialogs, the Cancel button is ambiguous because it could mean cancel the dialog or cancel the entire task.

- **If users need to both cancel the dialog box and the task, give commit buttons to do both.** Label the button that cancels the dialog box with a negative response to the main instruction. Label the button that cancels the entire task with Cancel. Using Cancel allows the dialog box to be used in many contexts.
- **If users just need to cancel the dialog but not the task, use a button with a specific, negative response to the main instruction,** and don’t have a Cancel button.

3.46 Command Links (Page 503)

- **Present a set of lengthy commands using command links, instead of command buttons or a combination of radio buttons and an OK button.** Doing so allows users to respond with a single click. However, this approach works only for a single question.
- **Present the most commonly used command links first.** The resulting order should roughly follow the likelihood of use, but also have a logical flow. **Exception:** Command links that result in doing everything should be placed first.
- If a command link requires further explanation, **provide a supplemental explanation.** Supplemental explanations describe why users might want to choose the command, or what happens if the command is chosen.
- **Don't use supplemental explanations that are wordy restatements of the command link.** Use a supplemental explanation only when you can't make a command link self-explanatory. Providing a supplemental explanation for one command link doesn't mean that you have to provide them for all commands.
- **Use phrases that start with a verb, without ending punctuation.**
- **If a command is strongly recommended, consider adding "(recommended)" to the label.** Be sure to add to the link label, not the supplemental explanation.
- **If a command is intended only for advanced users, consider adding "(advanced)" to the label.** Be sure to add to the link label, not the supplemental explanation.
- **Always provide an explicit Cancel button**. Don't use a command link for this purpose. **Incorrect:** *In this example, the dialog box uses a command link instead of a Cancel button.*

3.46.1 Don't Show This <Item> Again

- Consider using a *Don't show this <item> 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.
- Use this specific phrasing—replace <item> 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.
- Make the setting persist on a per-user basis.

- 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.
- If users may need to restore these dialog boxes, provide a Restore messages command in the program's Options dialog box.

3.46.2 Ask me later

- Provide this option to dismiss a dialog box only when: **The dialog box is indirect**, so users are likely to be focused on another task.
- **Users must respond but not immediately**, so they can continue with their work.
- **The question requires sufficient thought or effort** such that users might make better decisions if given more time.
- **The dialog box or option will be presented automatically later** (so that users really are asked later). Otherwise, expect users to respond now, but allow them to close the dialog box normally with either Cancel or Close. When used properly, this option should be rare.

3.46.3 More/Fewer

- Use More/Fewer progressive disclosure buttons to show or hide advanced or rarely used options, commands, or details that target users typically don't need. Doing so simplifies the dialog box for typical usage. Don't hide commonly used options, commands, or information because users might not find them.
- Don't use More/Fewer controls unless there really is more detail to show. Don't just restate the same information in a different format.
- Don't use More/Fewer controls to show Help. Use Help links or footnotes instead.
- With task dialogs, avoid combining More/Fewer controls with *Don't show this <item> again*. This combination has an awkward appearance.
- For labeling guidelines, see Progressive Disclosure.

3.46.4 Footnotes

Use footnotes for information that's not essential to a dialog box's purpose, but that users may find helpful in making decisions. Most users should be able to skip footnotes and still make informed decisions in their response to the dialog box.

3.46.5 Disabling or Removing Controls vs. Giving Error Messages

When a control doesn't apply in the current context, consider the following options:

- Remove the control when there is no way for users to enable it, or users don't expect it to apply and its state doesn't change frequently. Doing so simplifies the dialog box, and users won't miss it. Having a control appear and disappear frequently is annoying.
- Disable the control when users expect it to apply or its state changes frequently, and users can easily deduce why the control is disabled. An example of easy deduction is disabling a commit button when there is a single, empty text box that requires any input. You can use balloons to display non-critical user input problems with text boxes and editable drop-down lists. However, if the problem can't be explained with a balloon or involves multiple controls, the deduction would no longer be easy.
- Otherwise, leave the control enabled, but give an error message when it is used incorrectly. Disabling in this case would make it difficult for users to understand why the control is disabled. Users would be forced to determine the problem through experimentation and deductive logic. It's better just to provide a helpful error message to explain the problem explicitly.

  Tip: If you aren't sure whether you should disable a control or give an error message, start by composing the error message that you might give. If the error message contains helpful information that target users aren't likely to quickly deduce, leave the control enabled and give the error. Otherwise, disable the control.
- If you disable a control, also disable all associated controls, such as its label, supplemental explanations, or command buttons. However, don't disable its group box, group label, or group explanation if there are any.

3.46.6 Required Input

- To indicate that users must provide information in a control, consider the following options: **Don't indicate anything, but handle missing required input with error messages.** This approach reduces clutter and works well if most input is optional or users aren't likely to skip controls, thus keeping the number of error messages low.
- **Indicate required input using an asterisk at the beginning of the label.** Explain the asterisk using either:
  - A footnote at the bottom of the content area that says ** Required input*.
  - A tooltip on the asterisk that says *Required input*.

  This approach works well if there aren't many required controls, but poorly if most controls are required.
- **If all controls require input, state "All input required" at an appropriate place at the top of the content area.** This approach reduces clutter for this specific case.

- **Indicate optional inputs with "(optional)" after the label.** This approach works well if most input is required, but poorly otherwise.
- **For consistency, try to use the same method to indicate required input throughout your program.** Specifically, indicate either required or optional input as needed, but avoid using both within the same program.

3.46.7 Error Handling

- Prevent errors by using controls that are constrained to valid user input. You can also help reduce the number of errors by providing reasonable default values.
- Validate user input as soon as possible, and show errors as closely to the point of input as possible.
- Use modeless error handling (in-place errors or balloons) for user input problems. Use balloons for noncritical, single-point user input problems detected while in a text box or immediately after a text box loses focus. Balloons don't require available screen space or the dynamic layout that is required to display in-place messages. Display only a single balloon at a time. Because the problem isn't critical, no error icon is necessary. Balloons go away when clicked, when the problem is resolved, or after a timeout.
- Use in-place errors for delayed error detection, usually errors found by clicking a commit button. (Don't use in-place errors for settings that are immediately committed.) There can be multiple in-place errors at a time. Use normal text and a 16 × 16 pixel error icon, placing them directly next to the problem whenever possible. In-place errors don't go away unless the user commits and no other errors are found.
- Use modal error handling (task dialogs or message boxes) for all other problems, including errors that involve multiple controls, or are noncontextual or noninput errors found by clicking a commit button.
- When an input problem is found and reported, set input focus to the first control with the incorrect data. Scroll the control into view if necessary.

3.46.8 Help

When providing user assistance, consider the following options (listed in their order of preference):

- Give interactive controls self-explanatory labels. Users are more likely to read the labels on interactive controls than any other text.
- Provide in-context explanations using **static text labels**.
- Provide a specific Help link to a relevant Help topic.

- **Locate Help links at the bottom of the content area of the dialog box.** If the dialog box has a footnote and the Help link is related to it, place the Help link within the footnote.
- **Exception:** If a dialog box has several distinct groups of settings that have separate Help topics (perhaps within group boxes), locate the Help links at the bottom of the groups.
- **Don't use general or vague Help topic links or generic Help buttons.** Users often ignore generic Help.

3.46.9 Default Values

- Include a default commit button on every dialog box.
- For question dialogs:
  - Select the safest (to prevent loss of data or system access), most secure response to be the default. If safety and security aren't factors, select the most likely or convenient response.
  - **Exception:** Don't make a destructive response the default unless there is an easy, obvious way to undo the command.
- For choice dialogs:
  - For the initial default values, **select the safest (to prevent loss of data or system access) and most secure values for each control.** If safety and security aren't factors, select the most likely or convenient options.
  - For the subsequent default values, **reselect the previously selected options if those values are likely to be repeated, and doing so is safe and secure.** Otherwise, select the initial default values.

3.46.10 Recommended Sizing And Spacing

- **Support the minimum Windows Vista screen resolution of 800 × 600 pixels.** Layouts may be optimized for resizable windows using a screen resolution of 1024 × 768 pixels.
- **Use resizable windows whenever practical to avoid scroll bars and truncated data.** Windows with dynamic content and lists benefit the most from resizable windows.
- **Fixed-sized windows must be entirely visible and sized to fit within the work area.**
- **Resizable windows may be optimized for higher resolutions, but sized down as needed at display time to the actual screen resolution.**
- **Choose a default window size appropriate for its contents.** Don't be afraid to use larger initial window sizes if you can use the space effectively.

3.46.11 Text

3.46.11.1 General

- **Remove redundant text.** Look for redundant text in titles, main instructions, supplemental instructions, content areas, command links, and commit buttons. Generally, leave full text in instructions and interactive controls, and remove any redundancy from the other places.
- **Use positive phrasing.** Positive phrasing is easier for users to understand.
- **If necessary, use the word "window" to refer to the dialog box itself.**
- **Use the second person ("you/your") to tell users what to do** in the main instruction and content area. Often the second person is implied.
- **Use the first person ("I/me/my") to let users tell the program what to do** in controls in the content area that respond to the main instruction.

3.46.11.2 Dialog Box Titles

- **Use the title to identify the command, feature, or program where a dialog box came from.** If dialog is user initiated, identify it using the command or feature name. **Exceptions:**
  - If a dialog box is displayed by many different commands, consider using the program name instead.
  - If that title would be redundant with the main instruction, use the program name instead.
  - If it is program or system initiated (and therefore out of context), identify it using the program or feature name to give context.
- Don't use the title to explain what to do in the dialog—that's the purpose of the main instruction.
- Use the exact command name for command-based names, but don't include the ellipsis if there is one. You can include the command's menu title if necessary to compose a good title. Example: for an Object... command in an Insert menu, use the title *Insert Object*.
- **If a modeless dialog box appears on the taskbar, optimize the title for display on the taskbar** by concisely placing the distinguishing information first. Examples: "66% Complete," and "3 Reminders."
- **Don't include the words "dialog" or "progress" in the title.** This is implied, and leaving it off makes it easier for users to scan.
- Use **title-style capitalization**, without ending punctuation.

3.46.11.3 Main Instructions

- **Use the main instruction to explain concisely what to do in the dialog.** The instruction should be a specific statement, imperative direction, or question. Good instructions communicate the user's objective with the dialog rather than focusing purely on the mechanics of manipulating it.
- **Omit the main instruction when the only thing you can say is obvious.** In such cases, the content of the dialog box is self-explanatory. For example, the File Open and File Save common dialogs don't need a main instruction because their context and design make their purpose obvious.
- **Omit control labels that restate the main instruction.** In this case, the main instruction takes the access key.
- **Be concise—use only a single, complete sentence.** Pare the main instruction down to the essential information. If you must explain anything more, use supplemental instruction.
- **Use specific verbs whenever possible.** Specific verbs (examples: connect, save, install) are more meaningful to users than generic ones (examples: configure, manage, set).
- Use **sentence-style capitalization**.
- **Don't include final periods if the instruction is a statement.** If the instruction is a question, include a final question mark.
- **For progress dialogs, use a gerund phrase briefly explaining the operation in progress,** ending with an ellipsis. Example: Printing your pictures...

  Tip: You can evaluate a main instruction by imagining what you would say to a friend. If responding with the main instruction would be unnatural, unhelpful, or awkward, rework the instruction.

3.46.11.4 Supplemental Instructions

- When necessary, use an optional supplemental instruction to present additional information helpful to understanding or using the page. You can provide more detailed information and define terminology.
- If the appearance of the dialog box is program or system initiated (and therefore out of context), use the supplemental instruction to explain why the dialog has appeared. For such dialogs, the context is usually not obvious.
- Don't repeat the main instruction with slightly different wording. Instead, omit the supplemental instruction if there is not more to add.
- Use complete sentences, sentence-style capitalization, and ending punctuation.

3.46.11.5 Command Links

- **Choose concise link text that clearly communicates and differentiates what the command link does.** It should be self-explanatory and correspond to the main instruction. Users shouldn’t have to figure out what the link really means or how it differs from other links.
- **Always start command links with a verb.**
- Use sentence-style capitalization.
- Don’t use ending punctuation.
- **If necessary, provide any further explanation using complete sentences and ending punctuation.** However, add such explanations only when needed—don’t add explanations to all command links just because one command link needs one.

3.46.11.6 Commit Buttons

- **Use specific commit button labels that make sense on their own and are a response to the main instruction.** 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.
- **Start commit button labels with a verb. Exceptions are OK, Yes, and No.**
- Use sentence-style capitalization.
- Don’t use ending punctuation.
- Assign a unique **access key**. **Exception:** Don’t assign access keys to OK and Cancel buttons because Enter and Esc are their access keys. Doing so makes the other access keys easier to assign.

3.47 Dialog Box Design Concepts (Page 514)

Make sure that your dialog box design (determined by its purpose, type, and user interaction) matches its usage (determined by its context, probability of user action, and frequency of display).

Do the right thing by default. Don’t force users to configure their way out of a bad initial experience. Keep in mind that littering your program with unnecessary modal dialog boxes is more likely to foster outrage than user education. At a certain point, users tend to dismiss such dialog boxes without reading them.

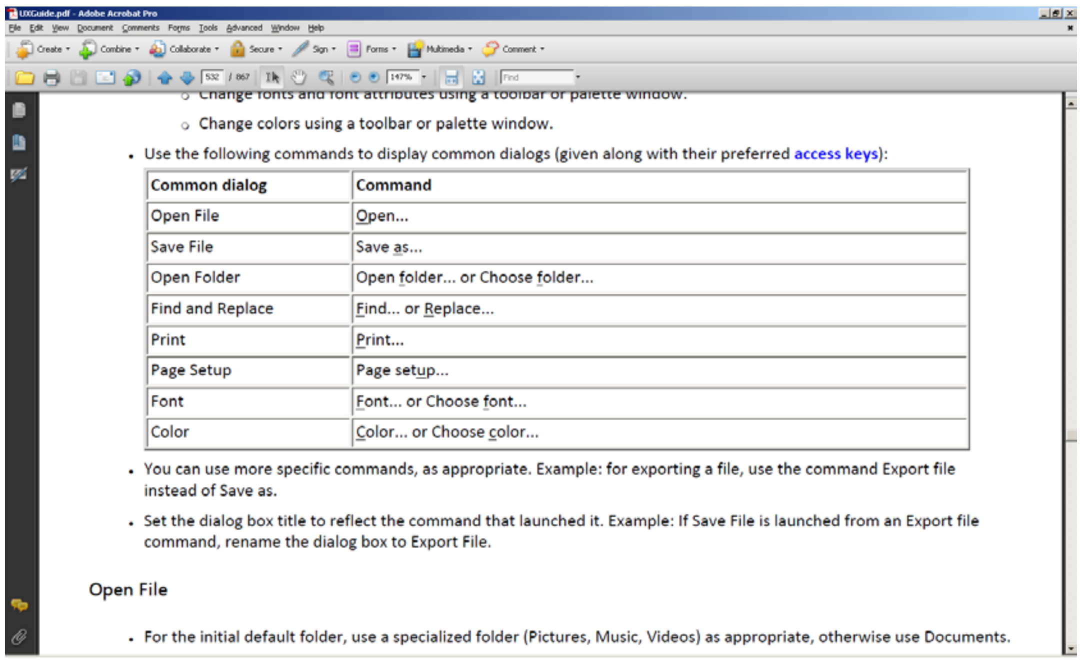
3.48 Common Dialogs (Page 525)

3.48.1 Guidelines

3.48.1.1 General

When appropriate, provide more direct or modeless alternatives. Allow users to:

- Open files by dropping them on your program.
- Save files using their current name and location with a Save command.
- Find the next occurrence of a string using the F3 key.
- Print one copy of an entire document to the default printer with a Print command.
- Change fonts and font attributes using a toolbar or palette window.
- Change colors using a toolbar or palette window.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
532 / 867
147%
Find

- Change fonts and font attributes using a toolbar or palette window.
- Change colors using a toolbar or palette window.

- Use the following commands to display common dialogs (given along with their preferred access keys):

Common dialog	Command
Open File	Open...
Save File	Save as...
Open Folder	Open folder... or Choose folder...
Find and Replace	Find... or Replace...
Print	Print...
Page Setup	Page setup...
Font	Font... or Choose font...
Color	Color... or Choose color...

- You can use more specific commands, as appropriate. Example: for exporting a file, use the command Export file instead of Save as.
- Set the dialog box title to reflect the command that launched it. Example: If Save File is launched from an Export file command, rename the dialog box to Export File.

Open File

- For the initial default folder, use a specialized folder (Pictures, Music, Videos) as appropriate, otherwise use Documents.

Figure 3-64: Common dialogs and commands

You can use more specific commands, as appropriate. Example: for exporting a file, use the command Export file instead of Save as.

Set the dialog box title to reflect the command that launched it. Example: If Save File is launched from an Export file command, rename the dialog box to Export File.

3.48.1.2 Open File

- For the initial default folder, use a specialized folder (Pictures, Music, Videos) as appropriate, otherwise use Documents.
- For subsequent default folders, use the last folder opened by the user using the program.
- When opening photo files, suppress file names by default. Photos are usually identified by their thumbnails and their names typically aren't meaningful.

3.48.1.3 Save File

- For the initial default folder (if a new file is being saved for the first time), use the specialized folder (Pictures, Music, Videos) as appropriate, otherwise use Documents.
- For temporary files, use the current user's temporary folder. Choose plain, but unique file names. Example: Use File0001.tmp instead of ~DF1A92.tmp.
- For the initial default file name, use a unique default name based on: The file's contents, if known. Example: The first words in a document.
- A pattern chosen by the user. Example: If the previous file was named "Hawaii 1.jpg", choose "Hawaii 2.jpg" as the next file.
- A generic pattern based on the file type. Example: "Photo1.jpg".
- For subsequent defaults (if the file already exists), use the file's current folder and name.
- When saving a file, preserve its creation date. If your program saves files by creating a temporary file, deletes the original, and renames the temporary file to the original file name, be sure to copy the creation date from the original file.
- Use Save File if the user selects the Save command without specifying a file name.

3.48.1.4 File Types Lists

Note: File types lists are used by Open File and Save File to determine the types of files displayed and the default file extension.

- If the file types list is short (five or fewer), order the list by likelihood of usage. If the list is long (six or more), use an alphabetical order to make the types easy to find.

- For Save File, include all variations of the supported file extensions, even if uncommon, and put the most common extension first. The file handling logic looks at this list to determine if the user supplied a supported file extension. Example: If a JPEG file types list includes only .jpg and .jpeg, the file test.jpe might be saved as test.jpe.jpg.
- For Save File, the initial default file type is the most likely chosen by the target user. The subsequent default is the file's current type.
- For Open File, the initial default file type is the most likely chosen by the target user. The subsequent default should be the last file type used.
- For Open File, include an "All files" entry as the first item if users can open any file type, or may need to see all files in a folder at the same time. Consider providing other meta filters, such as "All pictures," "All music," and "All videos." Place these immediately after "All files."
  - Use the format "File type name (*.ext1; *.ext2)." The file type name should be the registered file type name, which you can view in the Folder Options control panel item. Example: "HTML document (*.htm; *.html)." **Exception:** For meta-filters, remove the file extension list to eliminate clutter. Examples: "All files," "All pictures," "All music," and "All videos."
  - Use **sentence-style capitalization** for the file type names, and lowercase for the file type extensions.

3.48.1.5 Open Folder

For new programs, use the Open Files dialog in the "pick folders" mode. Doing so requires Windows Vista or later, so use the Open Folder dialog for programs that run in earlier versions of Windows. Developers: You can use the Open Files dialog in the "pick folders" mode by using the FOS_PICKFOLDERS flag.

3.48.1.6 Font

If necessary, you can filter the font list to show only the fonts available to your program.

3.48.1.7 Persistence

- Consider making the following values persistent to use as subsequent defaults: Input values (examples: default folders, default file names).
- Selected options (examples: selected printer, printing options).
- Views (examples: showing pictures in thumbnail view, showing pictures without file names, sorting by date, column widths).
- Presentation (examples: window size, location, and contents).
- **Exception:** Don't make these values persist for common dialogs when their usage is such that users are far more likely to want to start completely over.

- When determining default values, consider what target users are most likely to want based on the important scenarios. Also, consider scenarios within a program instance, across multiple instances (both consecutive or concurrent), and across multiple documents. Don't make values persist in circumstances that aren't likely to be helpful. **Example:** For a typical document-based application, it's helpful to use persistent Open File and Save File settings within a program instance and across consecutive instances, but keep concurrent instances independent. That way, users can work efficiently with several documents at a time.
- Make the settings persist on a per-program, per-user basis.

3.49 Wizards (Page 535)

No content

3.50 Property Windows (Page 536)

3.50.1 Guidelines

3.50.1.1 Property Sheets

- **Display a property sheet when users:** Select the Properties command for an object.
- Set input focus on an object and press Alt-Enter.

3.50.1.1.1 Multiple-Object Property Sheets

- **Display the common properties of all the selected objects.** Where the property values differ, display the controls associated with those values using a mixed state. (See the respective control guidelines for using mixed-state values.)
- If the selected object is a collection of multiple discrete objects (such as a file folder), **display the properties of the single grouped object instead of a multiple-object property sheet for the discrete objects.**

3.50.1.2 Options Dialog Boxes

Don't separate options from customization. That is, don't have both an Options command and a Customize command. Users are often confused by this separation. Instead, access customization through options.

3.50.1.3 Property Pages

- Follow these guidelines for page order:
  - Make the General page or its equivalent the first page.
  - Make the Advanced page or its equivalent the last page.

- For the remaining pages: Organize them into groups of related pages.
- Order the groups by the likelihood of their usage.
- Within each group, order the pages either by their relationships or by the likelihood of their use.

  You shouldn't have so many pages that there is a need to display them in alphabetical order.

- Make pages coherent by relating all properties on each page to a single, specific, task-based purpose.
- If space allows, explain the purpose of the property window at the top of the page if it isn't obvious to your target users. If the page is used to perform only a single task, phrase the text as a clear instruction about how to perform that task. Use complete sentences, ending with a period.
- Make similar content consistent across pages by using consistent control names and locations. For example, if several pages have Name boxes, try to place them in the same location on the page and use consistent labels. Similar content shouldn't bounce around from page to page.
- Place the same property on the same page throughout your application. For example, don't put an Expiration property on the General tab for one object type, and on the Advanced tab for another type.
- If users are likely to start with the last page displayed, make the page tab persist, and select it by default. Make the settings persist on a per-property window, per-user basis. Otherwise, select the first page by default.
- Don't make the settings on a page dependent upon settings on other pages. Put the dependent settings on a single page instead. Changing a setting on one page should never automatically change settings on other pages. Exception: If the dependent settings are in two different property windows, use static text labels to explain this relationship in both locations.
- Don't scroll property pages. Both tabs and scrollbars are used to increase the effective area of a window, but one mechanism should be sufficient. Instead of using scrollbars, make the property pages larger and lay out the pages efficiently.

3.50.1.4 First Pages

- For object properties, put the object's name on the first page.
- If you are associating (optional) icons with your objects, display the appropriate icon in the upper-left corner of the first page.

3.50.1.5 General Pages

- **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.

3.50.1.6 Advanced Pages

- **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 call properties advanced based solely on technological measures.** For example, a printer stapling option may be an advanced printer feature, but it is meaningful to all users, so it shouldn't be on an Advanced page.

3.50.1.7 Owned Property Windows

- **Don't display more than one owned property window from a property window.** Displaying more than one makes the meaning of the OK and Cancel buttons difficult to understand. You can display other types of auxiliary dialog boxes (such as object pickers) as needed.
- For property windows that use a delayed commit model, **make sure users can cancel changes made in an owned property window by clicking Cancel on the owner window.**
- If an owned property window requires an immediate commit, **indicate that changes were committed by renaming the Cancel button on the owner window to Close.** Revert the button back to Cancel if the user clicks Apply.

3.50.1.8 Other Owned Windows

If an owned window is used to perform an auxiliary task, **don't rename the Cancel button.** The preceding guidelines apply only to owned property windows, not dialog boxes used to perform auxiliary tasks.

3.50.2 Tabs

- **Use concise tab labels.** Use one or two words that clearly describe the content of the page. Longer labels result in an inefficient use of screen space, especially when the labels are localized.
- **Use specific, meaningful tab labels.** Avoid generic tab labels that could apply to any tab, such as General, Advanced, or Settings.

- **Use horizontal tabs if:**
  - The property window has seven or fewer tabs (including any third-party extensions).
  - **All the tabs fit on one row, even when the UI is localized.**
  - You use horizontal tabs on the other property windows in your application.
- Use vertical tabs if:
  - The property window has eight or more tabs (including any third-party extensions).
  - Using horizontal tabs would require more than one row.
  - You use vertical tabs on the other property windows in your application.
- **For property inspectors, to conserve space, consider using a drop-down list instead of tabs**, especially if the current tab is rarely changed by the user.
- **If a tab doesn't apply to the current context and users don't expect it to, remove the tab.** Doing so simplifies the UI, and users won't miss it.
- If a tab doesn't apply to the current context and users might expect it to:
  - Display the tab.
  - Disable the controls on the page.
  - Include text explaining why the controls are disabled.
  - Don't disable the tab because doing so isn't self-explanatory and prohibits exploration. Furthermore, users looking for a specific property would be forced to look on all other tabs.
- Don't assign effects to changing tabs. Changing the current tab should never have side effects, apply settings, or result in an error message.
- Don't nest tabs or combine horizontal tabs with vertical tabs. Instead, reduce the number of tabs, use only vertical tabs, or use another control such as a drop-down list.
- 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. Extensible property windows (which can be extended by third parties) always need to use tabs.
- Don't put icons on tabs. Icons usually add unnecessary visual clutter, consume screen space, and often don't improve user comprehension. Only add icons that aid in comprehension, such as standard symbols.
- Don't use product logos for tab graphics. Tabs aren't for branding.
- Don't scroll horizontal tabs. Horizontal scrolling isn't readily discoverable. You may scroll vertical tabs, however.

3.50.3 Command Buttons

- Place command buttons that apply to all property pages at the bottom of the property window. Right-align the buttons and use this order (from left to right): OK, Cancel, and Apply.
- Place command buttons that apply only to individual property pages directly on the property page.

3.50.4 Commit buttons

3.50.4.1 OK Buttons

- For owner property windows, the OK button means apply the pending changes (made since the window was opened or the last Apply), and close the window.
- For owned property windows, the OK button means keep the changes, close the window, and apply the changes when the owner window's changes are applied.
- Don't rename the OK button. Unlike other dialog boxes, property windows aren't used to perform any one specific task. If it makes sense to rename the OK button (to Print, for example), the window isn't a property window.
- Don't assign an access key.

3.50.4.2 Cancel Buttons

- The Cancel button means discard all pending changes (made since the window was opened or the last Apply), and close the window.
- If all pending changes can't be abandoned, rename the Cancel button to Close. Clicking Cancel must abandon all pending changes.
- If the owned property window requires an immediate commit, rename the Cancel button on the owner window to Close to show that changes were committed.
- Don't assign an access key.

3.50.4.3 Apply Buttons

- For owner property sheets, the Apply button means apply the pending changes (made since the window was opened or the last Apply), but leave the window open. Doing so allows users to evaluate the changes before closing the property sheet.
- For owned property sheets, don't use. Using an Apply button on an owned property sheet makes the meaning of the commit buttons on the owner property sheet difficult to understand.

- Provide an Apply button only if the property sheet has settings (at least one) with effects that users can evaluate in a meaningful way. Typically, Apply buttons are used when settings make visible changes. Users should be able to apply a change, evaluate the change, and make further changes based on that evaluation. If not, remove the Apply button instead of disabling it.
- Place all settings that users may want to apply on owner pages. Don't use Apply buttons on owned property sheets, because doing so is confusing.
- Use Apply buttons only with property sheets, not with options dialog boxes.
- Enable the Apply button only when there are pending changes; otherwise, disable it.
- Assign "A" as the access key.

3.50.4.4 Close Buttons

- If all pending changes can't be abandoned, rename the Cancel button to Close. Clicking Cancel must abandon all pending changes.
- Don't confirm if users discard their changes. Exception: If the property window has settings that require significant effort to set and the user has made changes, you may display a confirmation if the user clicks the Close button on the title bar. The reason is that some users mistakenly assume that the Close button on the title bar has the same effect as the OK button.
- With the exception of the confirmation message, make sure the Close button on the title bar has the same effect as Cancel or Close.

3.50.5 Page Contents

- **Make sure the properties are necessary.** Don't clutter your pages with unnecessary properties just to avoid making hard design decisions.
- **Present properties in terms of user goals, not 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 how the user benefits from that setting.
- **Present properties at the right level.** You don't need to present individual, low-level settings on a property page, so present the properties at a level that makes sense to your users.
- **Design property pages for specific tasks.** Determine the tasks that users will perform, and make sure there is a clear path to perform those tasks.
- **Organize property pages efficiently** by reducing the number of tabs, deciding what goes on a page based on logical grouping and coherence, and simplifying the page's presentation.

- If an option is strongly recommended, consider adding "(recommended)" to the label.
- Provide a Restore Defaults command button for a property page or the entire property window when:
  - Your users are likely to consider the settings complex and difficult to understand.
  - Having incorrect settings may result in breaking functionality, but the defaults might restore functionality.
  - It's easier for users to start over when the object is misconfigured.
  - Confirm the Restore Defaults command if its effect isn't obvious or the settings are complex. Indicate the confirmation by using **ellipses**.
- When appropriate, display a preview of the results of a setting.

3.50.6 Help

When providing user assistance, **consider using the following options** (listed in their order of preference):

- Give interactive controls self-explanatory labels. Users are more likely to read the labels on interactive controls than any other text.
- Provide in-context explanations using static text labels.
- Provide a specific **link** to a relevant Help topic.
- **Locate Help links at the bottom of each page.** If a page has several distinct groups of settings that have a Help topic (perhaps within group boxes), locate the Help link at the bottom of the group.
- **Don't use general or vague Help topic links or generic Help buttons.** Users often ignore generic Help.

3.50.7 Standard Users And Protected Administrators

Many settings require administrator privileges to change. If a process requires administrator privileges, Windows® and later requires **Standard users** and **Protected administrators** to elevate their privileges explicitly. Doing so helps prevent malicious code from running with administrator privileges.

3.50.8 Default Values

- **The settings within a property window must reflect the current state of the application, object, or collection of objects.** Doing otherwise would be misleading and possibly lead to undesired results. For example, if the settings reflect the recommendations but not the current state, users might click Cancel instead of making changes, thinking that no changes are needed.

- **Choose the safest (to prevent loss of data or system access) and most secure initial state.** Assume that most users won't change the settings.
- **If safety and security aren't factors, choose the initial state that is most likely or convenient.**

3.51 Text (Page 551)

3.51.1 Commands

- To display program options, use "Options."
- To display an object's property window, use "Properties."
- To display a summary of the commonly used program customization settings, use "**Personalize**."
- Don't use "Settings" or "Preferences."
- Don't use **ellipses** for these commands.

3.51.2 Property Sheet Titles

- For a single object, use "[object name] Properties." If the object has no name, use the object's type name. (For example, User Account Properties.)
- For multiple objects, use "[first object name], ... Properties." If the objects have no names, use the objects' type name. (For example, User Accounts Properties.)
- If the objects have different types, use "Selection Properties."
- Use **title-style capitalization**.
- Don't use ending punctuation.
- Don't use hyphens, such as "[object name] - Properties."

3.51.3 Property Inspector Titles

- Use "Properties."
- Use title-style capitalization.
- Don't use ending punctuation.

3.51.4 Options Dialog Box Titles

- Use "Options."
- Use title-style capitalization.
- Don't use ending punctuation.

3.51.5 Property Page Tab Names

- **Use concise tab labels.** Use one or two words that clearly describe the content of the page. Using longer tab names results in an inefficient use of screen space, especially when the tab names are localized.
- **Use specific, meaningful tab labels.** Avoid generic tab labels that could apply to any tab, such as General, Advanced, or Settings.
- Write the label as a one- or two-word phrase and use no ending punctuation.
- Use **sentence-style capitalization**.
- Don’t assign a unique **access key**.

3.51.6 Property Page Text

- Avoid large blocks of text.
- Provide enough room for the text to expand 30% if it will be localized.
- Don’t use text phrased as a command on property windows. Because users might want to simply view settings, you don’t want to prompt them to change settings.
- Use sentence-style capitalization and ending punctuation.

3.52 Property Window Design Concepts (Page 553)

To ensure that your property windows are useful and usable, follow these steps:

- Make sure the properties are necessary.
- Present properties in terms of user goals, not technology.
- Present properties at the right level.
- Design pages for specific tasks.
- Design pages for Standard users and Protected administrators.
- Organize the property pages efficiently.

3.53 Visuals (Page 556)

3.53.1 Guidelines

3.53.1.1 Screen Resolution And Dpi

- **Support the minimum Windows effective resolution of 800 × 600 pixels.** For critical UIs that must work in safe mode, support an effective resolution of 640 × 480 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 1024 × 768 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 800 × 600 pixels), 120 dpi (at 1024 × 768 pixels), and 144 dpi (at 1200 × 900 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.

3.53.1.2 Window Size

- **Choose a default window size appropriate for its contents.** Don’t be afraid to use larger initial window sizes if you can use the space effectively.
- **Use a balanced height to width aspect ratio.** An aspect ratio between 3:5 and 5:3 is preferred, although an aspect ratio of 1:3 can be used for message dialog boxes (such as errors and warnings).
- **Use resizable windows whenever practical to avoid scroll bars and truncated data.** Windows with dynamic content, documents, images, lists, and trees benefit the most from resizable windows.
- **For text documents, consider a maximum line length of 80 characters** to make the text easy to read. (Characters include letters, punctuation, and spaces.)
- Fixed-sized windows: **Fixed-sized windows must be entirely visible and sized to fit within the work area.**
- Resizable windows: **Resizable windows may be optimized for higher resolutions, but sized down as needed at display time to the actual screen resolution.**
- **Progressively larger window sizes must show progressively more information.** Make sure that at least one window portion or control has resizable content.
- **Keep the upper-left origin of the content fixed as the window is resized.** Don’t move the origin to balance the content as the window size changes.
- **Set a maximum content size if the content can be too stretched too wide.** If the content becomes unwieldy, don’t resize the content area beyond its maximum width or change the content’s origin as the window is resized larger. Instead, keep a maximum width and a fixed upper-left origin.
- **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. Optional UI elements should be removed completely.

- **Consider altering the presentation to make the content usable at smaller sizes.**

3.53.1.3 Control Size

- **Make all interactive controls at least relative 16 × 16 pixels.** Doing so works well for all input devices, including Windows Tablet and Touch Technology.
- **Size controls to avoid truncated data.** Don’t truncate data that must be read to perform a task. Size list view columns to avoid truncated data.
- **Size controls to eliminate unnecessary scrolling.** Make controls slightly larger if doing so eliminates a scrollbar. There should be few vertical scrollbars and no unnecessary horizontal scrollbars.
- **Reduce the number of control sizes on a surface.** Prefer using the **standard recommended control sizes** and when necessary, use a few consistently sized larger or smaller controls. Try to use a single width for list boxes and tree views, and no more than three widths for command buttons and drop-down lists. However, text box and combo box widths should suggest the length of their longest or expected input.
- **For controls that are sized based on their text, include an additional 30% (up to 200% for shorter text) for any text that will be localized.** This guideline assumes the layout is designed using English text. Note also that this guideline refers to localized text, not numbers.
- **Extend static text controls, check boxes, and radio buttons to the maximum width that will fit in the layout.** Doing so avoids truncation from variable length text and localization.

3.53.1.4 Control Spacing

If controls aren’t touching, have at least 3 DLUs (5 relative pixels) of space between them. Otherwise, users may click on inactive space between the controls. Since clicking inactive space has no result or visual feedback, users are often uncertain what went wrong.

3.53.1.5 Placement

- Arrange the UI elements within a surface to flow naturally in a left-to-right, top-to-bottom order (in Western cultures). The placement of the UI elements conveys their relationship and should mirror the steps to perform the task.
- Place UI elements that initiate a task in the upper-left corner or upper-center. Give the UI element that users should look at first the greatest visual emphasis.
- Place UI elements that complete a task in the lower-right corner.
- Place related UI elements together, and separate unrelated elements.

- Place required steps in the main flow.
- Place optional steps outside the main flow, possibly deemphasized by using a suitable background or progressive disclosure.
- Place frequently used elements before infrequently used elements in the scan path.

3.53.1.6 Focus

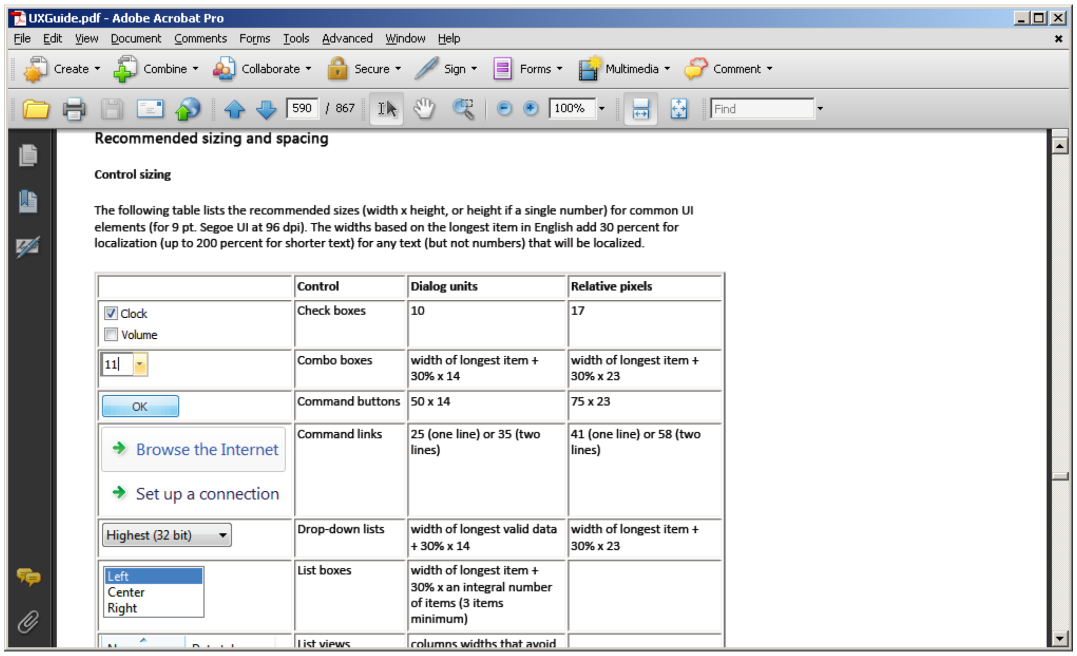
- **Choose a single UI element that users need to look at first to be the focal point.** The focal point should be something important that users need to find and understand quickly.
- **Place the focal point in the upper-left corner or upper-center.**
- **Give the focal point the greatest visual emphasis,** such as prominent text, default selection, or initial input focus.

3.53.1.7 Alignment

- Normally, use left alignment.
- Use right alignment for numeric data, especially columns of numeric data.
- Use right alignment for commit buttons, as well as controls aligned with right window edge.
- Use center alignment when either left or right alignment is inappropriate or appears unbalanced.
- When vertically aligning controls with text, align the text baselines to give a smooth horizontal reading flow.
- For label alignment, refer to the **Label alignment** section in Design concepts.

3.53.1.8 Accessibility

- **Don’t use layout as the only means to convey important information about a UI.** Users who have visual impairments may not be able to interpret this presentation. For example, make sure that controls labels communicate their relationship to other items.
- **Don’t embed subordinate controls within control labels to create a sentence or phrase.** Such associations are based purely on layout and aren’t handled well by keyboard navigation or accessibility assistive technologies. Furthermore, this technique isn’t localizable because sentence structure varies with language.
- **Make grouping accessible.** Groups defined by window panes, group boxes, separators, text labels, and aggregators are automatically handled by accessibility aids. However, groups defined only by placement and backgrounds are not, and must be defined programmatically for accessibility.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
590 / 867
100%
Find

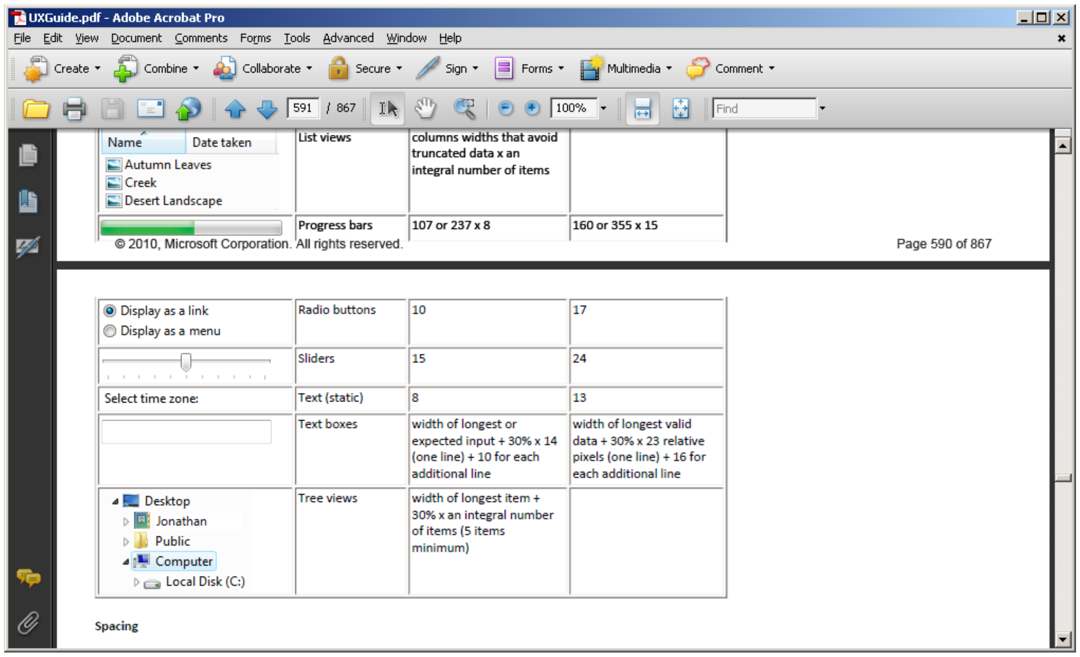
Recommended sizing and spacing

Control sizing

The following table lists the recommended sizes (width x height, or height if a single number) for common UI elements (for 9 pt. Segoe UI at 96 dpi). The widths based on the longest item in English add 30 percent for localization (up to 200 percent for shorter text) for any text (but not numbers) that will be localized.

	Control	Dialog units	Relative pixels
☑ Clock ☐ Volume	Check boxes	10	17
11	Combo boxes	width of longest item + 30% x 14	width of longest item + 30% x 23
OK	Command buttons	50 x 14	75 x 23
Browse the Internet / Set up a connection	Command links	25 (one line) or 35 (two lines)	41 (one line) or 58 (two lines)
Highest (32 bit)	Drop-down lists	width of longest valid data + 30% x 14	width of longest item + 30% x 23
Left Center Right	List boxes	width of longest item + 30% x an integral number of items (3 items minimum)	
Name Date taken / Autumn Leaves / Creek / Desert Landscape	List views	columns widths that avoid truncated data x an integral number of items	
	Progress bars	107 or 237 x 8	160 or 355 x 15

Figure 3-65: Recommended sizing and spacing: Control Sizing

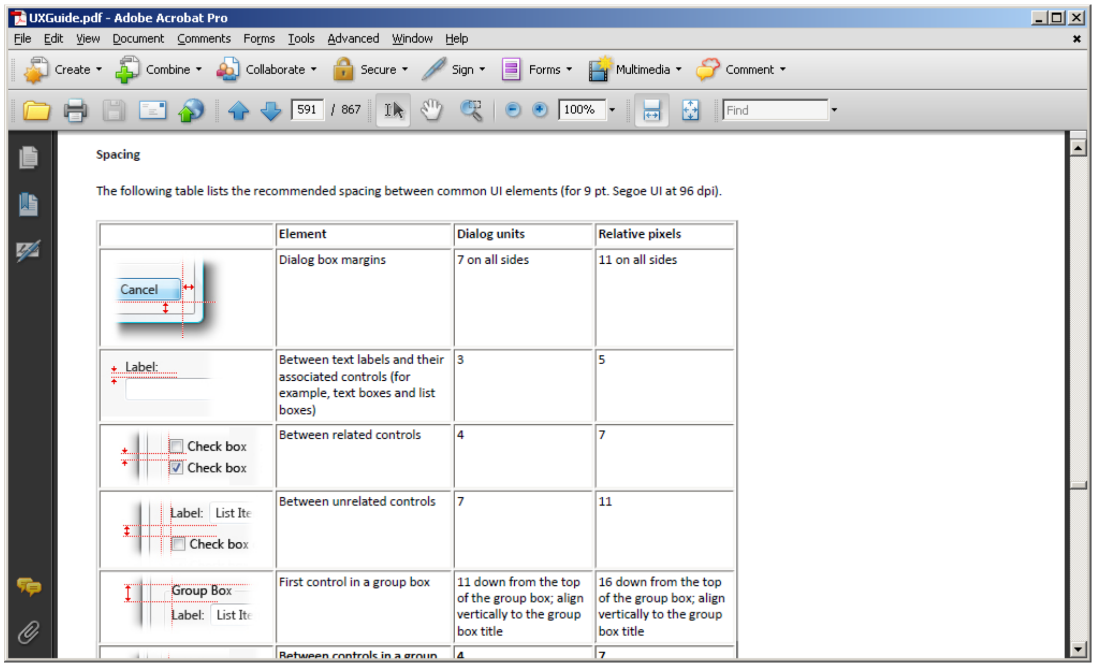

UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
591 / 867
100%
Find

© 2010, Microsoft Corporation. All rights reserved. Page 590 of 867

● Display as a link ○ Display as a menu	Radio buttons	10	17
	Sliders	15	24
Select time zone:	Text (static)	8	13
	Text boxes	width of longest or expected input + 30% x 14 (one line) + 10 for each additional line	width of longest valid data + 30% x 23 relative pixels (one line) + 16 for each additional line
Desktop / Jonathan / Public / Computer / Local Disk (C:)	Tree views	width of longest item + 30% x an integral number of items (5 items minimum)	

Spacing

Figure 3-66: Spacing

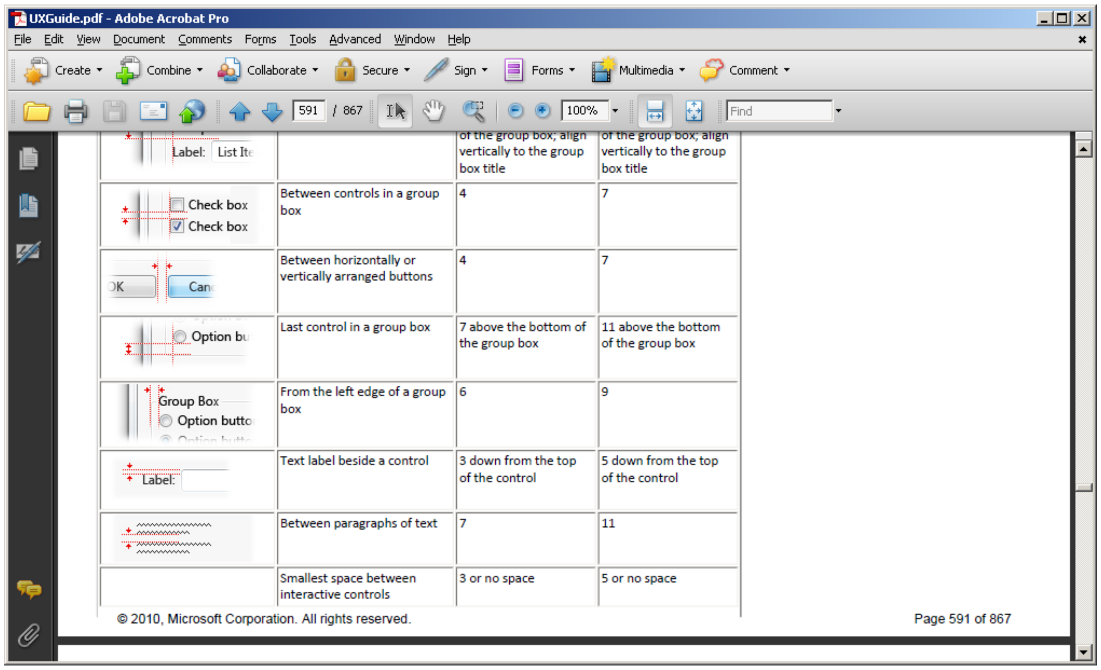

UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
591 / 867
100%
Find

Spacing

The following table lists the recommended spacing between common UI elements (for 9 pt. Segoe UI at 96 dpi).

	Element	Dialog units	Relative pixels
	Dialog box margins	7 on all sides	11 on all sides
	Between text labels and their associated controls (for example, text boxes and list boxes)	3	5
	Between related controls	4	7
	Between unrelated controls	7	11
	First control in a group box	11 down from the top of the group box; align vertically to the group box title	16 down from the top of the group box; align vertically to the group box title
	Between controls in a group	4	7

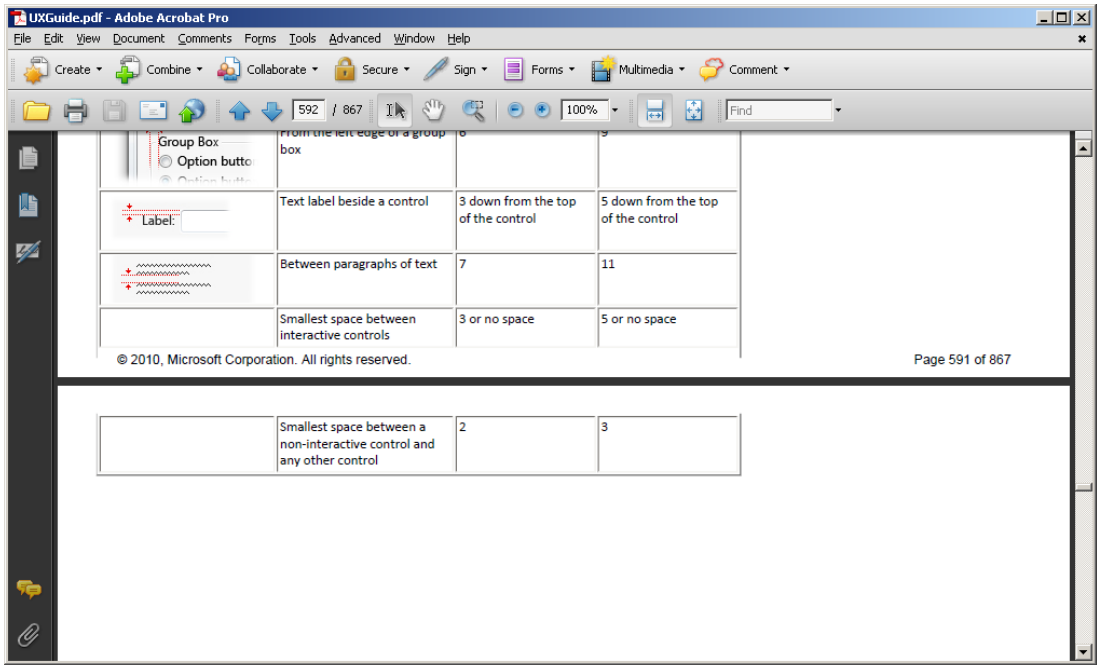
Figure 3-67: Spacing


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
591 / 867
100%
Find

	Element	Dialog units	Relative pixels
		of the group box; align vertically to the group box title	of the group box; align vertically to the group box title
	Between controls in a group box	4	7
	Between horizontally or vertically arranged buttons	4	7
	Last control in a group box	7 above the bottom of the group box	11 above the bottom of the group box
	From the left edge of a group box	6	9
	Text label beside a control	3 down from the top of the control	5 down from the top of the control
	Between paragraphs of text	7	11
	Smallest space between interactive controls	3 or no space	5 or no space

© 2010, Microsoft Corporation. All rights reserved. Page 591 of 867

Figure 3-68: Spacing


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
592 / 867
100%
Find

Group Box Option butto	From the left edge of a group box	6	9
Label:	Text label beside a control	3 down from the top of the control	5 down from the top of the control
	Between paragraphs of text	7	11
	Smallest space between interactive controls	3 or no space	5 or no space

© 2010, Microsoft Corporation. All rights reserved. Page 591 of 867

	Smallest space between a non-interactive control and any other control	2	3

Figure 3-69: Spacing

3.54 Window Frames (Page 598)

3.54.1 Guidelines

3.54.1.1 Window Frames

Use standard window frames. **Exception:** To give immersive full screen, stand-alone applications a unique feel:

- Consider hiding the window frame of the **primary window**.
- Consider using custom frames for **secondary windows**.
  – If a custom frame is appropriate, choose a design that is lightweight and doesn't draw too much attention to itself.
- Don't add controls to a window frame. Put the controls within the window instead.

3.54.1.2 Full Screen Mode

- For programs that have an optional full screen mode, to enable full screen mode:
  – Have a modal full screen command in the menu bar or toolbar. When the user clicks the command, show the command in its selected state.
  – Use F11 for the full screen shortcut key.

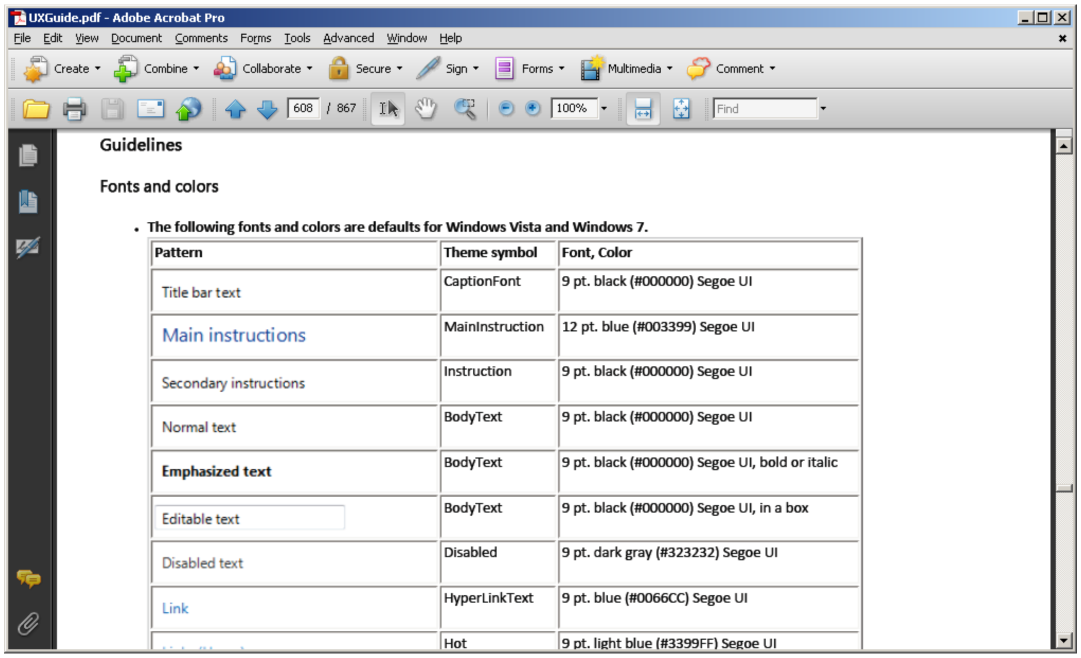
- If there is a toolbar and full screen mode is commonly used, also have a graphic toolbar button with a *Full screen* tooltip.
- To revert back from full screen mode: Have a modal full screen command in the menu bar or toolbar. When the user clicks the command, show the command in its cleared state.
- Use F11 for the full screen shortcut key. If not already assigned, Esc can also be used for this purpose.

3.54.1.3 Glass

Standard window frames use glass automatically in Windows, but you can also use glass in regions that touch the window frame.

- **Consider using glass strategically in small regions touching the window frame without text.** Doing so can give a program a simpler, lighter, more cohesive look by making the region appear to be part of the frame.
- **Don't use glass in situations where a plain window background would be more attractive or easier to use.**

3.55 Fonts (Page 605)


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
608 / 867 100% Find

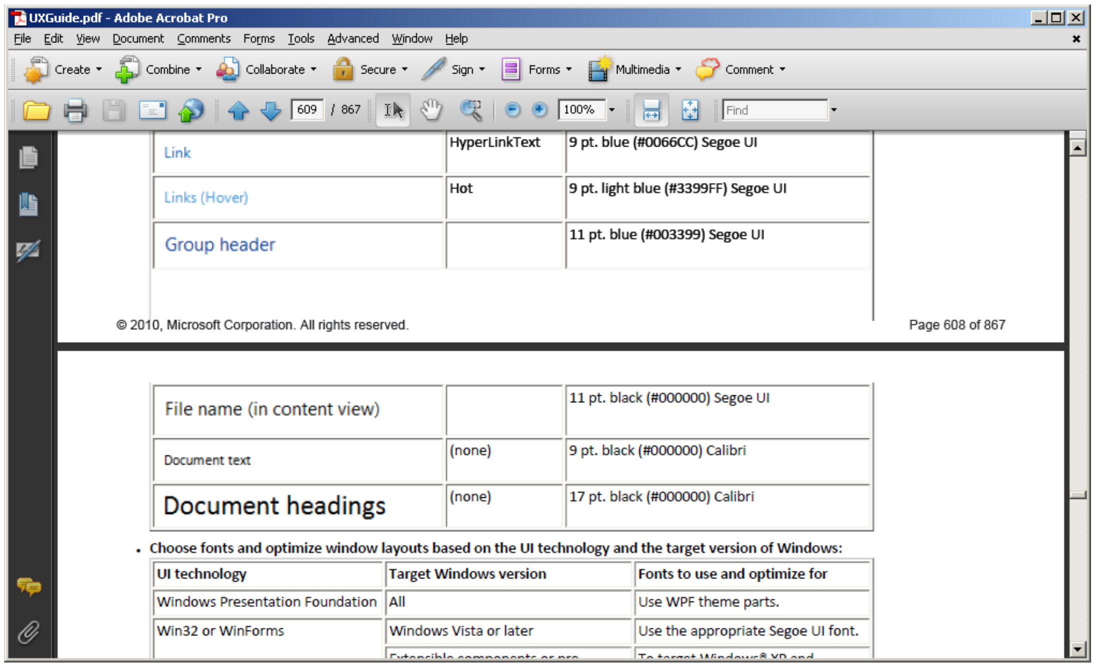
Guidelines

Fonts and colors

- The following fonts and colors are defaults for Windows Vista and Windows 7.

Pattern	Theme symbol	Font, Color
Title bar text	CaptionFont	9 pt. black (#000000) Segoe UI
Main instructions	MainInstruction	12 pt. blue (#003399) Segoe UI
Secondary instructions	Instruction	9 pt. black (#000000) Segoe UI
Normal text	BodyText	9 pt. black (#000000) Segoe UI
Emphasized text	BodyText	9 pt. black (#000000) Segoe UI, bold or italic
Editable text	BodyText	9 pt. black (#000000) Segoe UI, in a box
Disabled text	Disabled	9 pt. dark gray (#323232) Segoe UI
Link	HyperLinkText	9 pt. blue (#0066CC) Segoe UI
[illegible]	Hot	9 pt. light blue (#3399FF) Segoe UI

Figure 3-70: Guidelines: Fonts and colors


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
609 / 867
100%
Find

Link	HyperLinkText	9 pt. blue (#0066CC) Segoe UI
Links (Hover)	Hot	9 pt. light blue (#3399FF) Segoe UI
Group header		11 pt. blue (#003399) Segoe UI

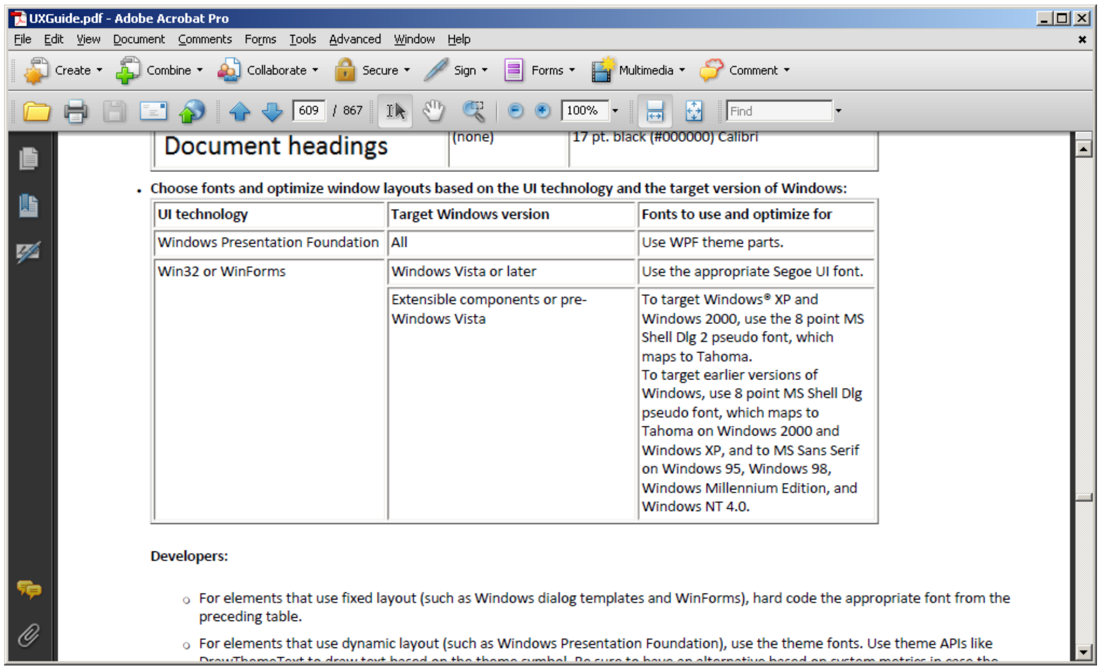
© 2010, Microsoft Corporation. All rights reserved. Page 608 of 867

File name (in content view)		11 pt. black (#000000) Segoe UI
Document text	(none)	9 pt. black (#000000) Calibri
Document headings	(none)	17 pt. black (#000000) Calibri

- Choose fonts and optimize window layouts based on the UI technology and the target version of Windows:

UI technology	Target Windows version	Fonts to use and optimize for
Windows Presentation Foundation	All	Use WPF theme parts.
Win32 or WinForms	Windows Vista or later	Use the appropriate Segoe UI font.

Figure 3-71: Document headings


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
609 / 867
100%
Find

Document headings	(none)	17 pt. black (#000000) Calibri

- Choose fonts and optimize window layouts based on the UI technology and the target version of Windows:

UI technology	Target Windows version	Fonts to use and optimize for
Windows Presentation Foundation	All	Use WPF theme parts.
Win32 or WinForms	Windows Vista or later	Use the appropriate Segoe UI font.
	Extensible components or pre-Windows Vista	To target Windows® XP and Windows 2000, use the 8 point MS Shell Dlg 2 pseudo font, which maps to Tahoma. To target earlier versions of Windows, use 8 point MS Shell Dlg pseudo font, which maps to Tahoma on Windows 2000 and Windows XP, and to MS Sans Serif on Windows 95, Windows 98, Windows Millennium Edition, and Windows NT 4.0.

Developers:

- For elements that use fixed layout (such as Windows dialog templates and WinForms), hard code the appropriate font from the preceding table.
- For elements that use dynamic layout (such as Windows Presentation Foundation), use the theme fonts. Use theme APIs like

Figure 3-72: Document headings

- **Developers:** For elements that use fixed layout (such as Windows dialog templates and WinForms), hard code the appropriate font from the preceding table.

- For elements that use dynamic layout (such as Windows Presentation Foundation), use the theme fonts. Use theme APIs like DrawThemeText to draw text based on the theme symbol. Be sure to have an alternative based on system metrics in case the theme service isn't running.
- **For Segoe UI, use a 9 point font size or larger.** The Segoe UI font is optimized for these sizes, so avoid using smaller sizes.
- **Always match system text colors with their corresponding background colors.** For example, if you choose COLOR_STATICTEXT for the text color, you must also choose COLOR_STATIC for the background color.
- **Always create new fonts based on proportional-sized variations of the system font.** Given the system font metrics, you can create bold, italic, larger, and smaller variations.
- Display large blocks of read-only text (such as license terms) against a light background instead of a gray background. Gray backgrounds suggest that the text is disabled and discourages reading.
- **Consider a maximum line length of 65 characters** to make the text easy to read. (Characters include letters, punctuation, and spaces.)

3.55.1 Attributes

- Most UI text should be plain—without any attributes. Attributes may be used as follows:
  - **Bold.** Use in control labels to make the text easier to parse. Use sparingly to draw attention to text users must read. Using too much bold lessens its impact.
  - **Italic.** Use to refer to text literally instead of quotation marks. Use sparingly to emphasize specific words. Use for **prompts** in **text boxes** and **editable drop-down lists**.
  - **Bold italic.** Don't use.
  - **Underline.** Don't use except for links. Use italic instead for emphasis.

Not all fonts support bold and italic, so they should never be crucial to understanding the text.

3.56 Color (Page 610)

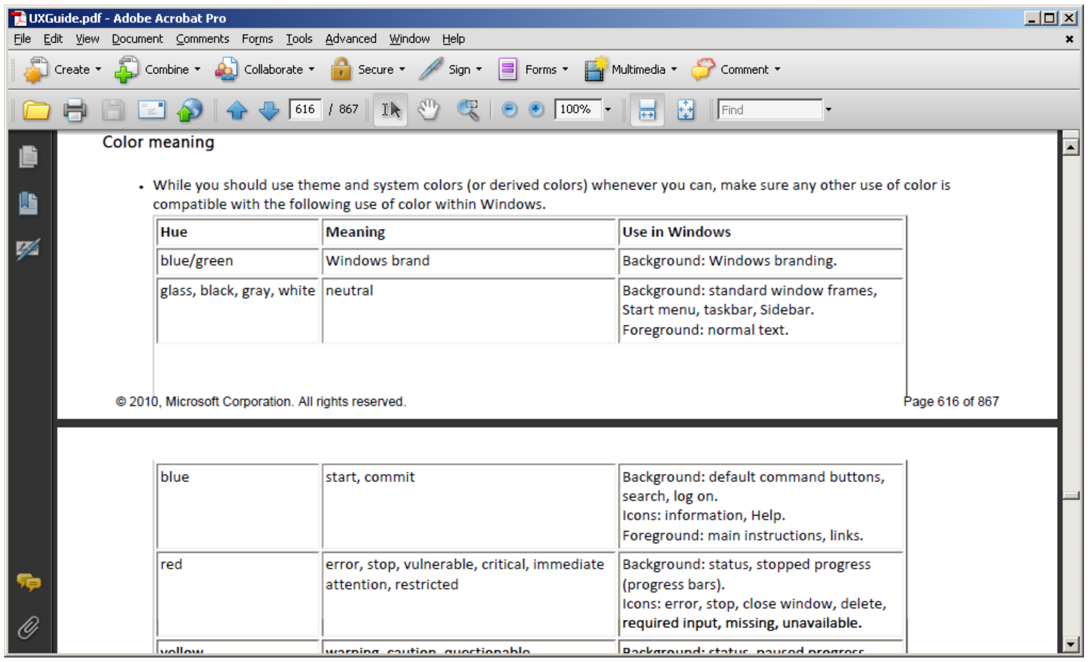
3.56.1 Guidelines

3.56.1.1 General

Never use color as a primary method of communication, but as a secondary method to reinforce meaning visually.

3.56.1.2 Using Theme And System Colors

- Whenever possible, **choose colors by selecting the appropriate theme color or system color.** By doing so, you can always respect users' color preference.
- **Choose theme and system colors based on their purpose.** Don't choose colors based on their current appearance, as that appearance can be changed by the user or future versions of Windows.
- **Match foreground colors with their associated background colors.** Foreground colors are guaranteed to be legible only against their associated background colors. Don't mix and match foreground colors with other background colors, or worse yet, other foreground colors.
- **Don't mix color types.** That is, always match theme colors with their associated theme colors, system colors with their associated system colors, and hardwired colors with other hardwired colors. For example, a theme text color isn't guaranteed to be legible against a hardwired background.
  - If you must use a color that isn't a theme or system color: Prefer to derive the color from a theme or system color over hardwiring its value. Use the process described earlier in this article, in Using other colors.
  - Handle high-contrast mode as a special case.
  - Handle theme changes. Theme changes are handled automatically by windows with standard window frames and common controls. Windows with custom window frames, custom or owner-draw controls, and other use of color must handle theme changes explicitly.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
616 / 867
100%
Find

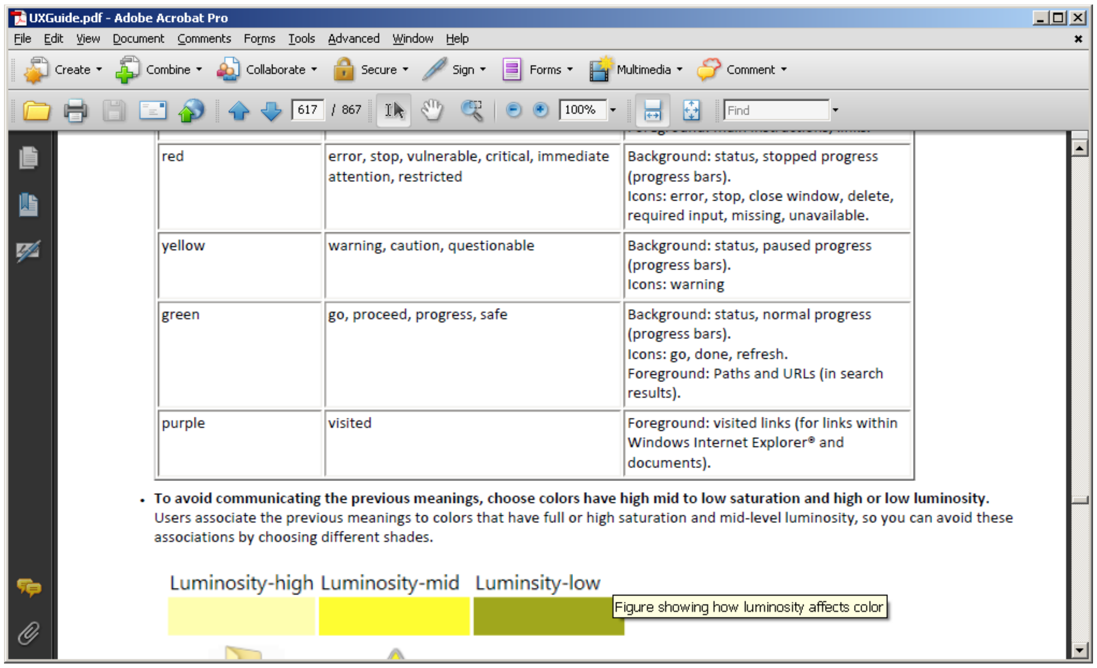
Color meaning

- While you should use theme and system colors (or derived colors) whenever you can, make sure any other use of color is compatible with the following use of color within Windows.

Hue	Meaning	Use in Windows
blue/green	Windows brand	Background: Windows branding.
glass, black, gray, white	neutral	Background: standard window frames, Start menu, taskbar, Sidebar. Foreground: normal text.
blue	start, commit	Background: default command buttons, search, log on. Icons: information, Help. Foreground: main instructions, links.
red	error, stop, vulnerable, critical, immediate attention, restricted	Background: status, stopped progress (progress bars). Icons: error, stop, close window, delete, required input, missing, unavailable.
yellow	warning, caution, questionable	Background: status, paused progress

© 2010, Microsoft Corporation. All rights reserved. Page 616 of 867

Figure 3-73: Color meaning


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
617 / 867
100%
Find

red	error, stop, vulnerable, critical, immediate attention, restricted	Background: status, stopped progress (progress bars). Icons: error, stop, close window, delete, required input, missing, unavailable.
yellow	warning, caution, questionable	Background: status, paused progress (progress bars). Icons: warning
green	go, proceed, progress, safe	Background: status, normal progress (progress bars). Icons: go, done, refresh. Foreground: Paths and URLs (in search results).
purple	visited	Foreground: visited links (for links within Windows Internet Explorer® and documents).

- **To avoid communicating the previous meanings, choose colors have high mid to low saturation and high or low luminosity.** Users associate the previous meanings to colors that have full or high saturation and mid-level luminosity, so you can avoid these associations by choosing different shades.

Luminosity-high Luminosity-mid Luminsity-low
Figure showing how luminosity affects color

Figure 3-74: Luminosity

3.56.1.3 Using Color In Data

- When helpful, **assign color to data to help users differentiate it.** Note that users will assume that data with similar colors have similar meanings.
- **Assign colors by default that are easy to distinguish.** Generally, colors are easy to distinguish if they are far apart from each other in the HSL/HSV color spaces, while maintaining high contrast with their background: When choosing colors, prefer triad harmonies or complementary hues, but not adjacent hues.

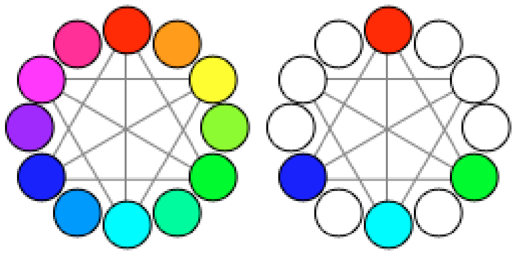


Figure 3-75: The color wheel and triad harmonies

  - Colors have high contrast if there is a large difference in their hue, saturation, or luminosity.
  - Using a white or very light background makes contrasting foreground colors easier to distinguish.
- **Allow users to customize these color assignments** because color choice is subjective and a personal preference. If there are many coordinated colors, allow users to change them as a group using color schemes.

- **Allow users to label these color assignments.** Doing so helps make them easier to identify and find.
- **Unlike UI colors, data should not change when the system colors change.**

3.56.2 Documentation

- **Refer to UI elements by their names, not by their colors.** Such references aren't accessible and the system colors may change. If a UI element's name isn't well known or not descriptive enough, show a screenshot to clarify.

3.57 Icons (Page 620)

3.57.1 Guidelines

3.57.1.1 Perspective

- Icons in Windows Vista are either three-dimensional and shown in perspective as solid objects, or two-dimensional objects shown straight-on. Use flat icons for files and for objects that are actually flat, like documents or pieces of paper.
- Three-dimensional objects are represented in perspective as solid objects, seen from a low birds-eye view with two vanishing points.
- In the smaller sizes, the same icon may change from perspective to straight-on. At the size of 16 × 16 pixels and smaller, render icons straight-on (front-facing). For larger icons, use perspective. Exception: Toolbar icons are always front-facing, even in larger sizes.

3.57.1.2 Light Source

- The light source for objects within the perspective grid is above, slightly in front of, and slightly to the left of the object.
- The light source casts shadows that are slightly to the rear and right of the object's base.
- All light rays are parallel, and strike the object along the same angle (like the sun). The goal is to have a uniform lighting appearance across all icons and spotlight effects. Parallel light rays produce shadows that all have the same length and density, providing further unity across multiple icons.

3.57.2 Shadows

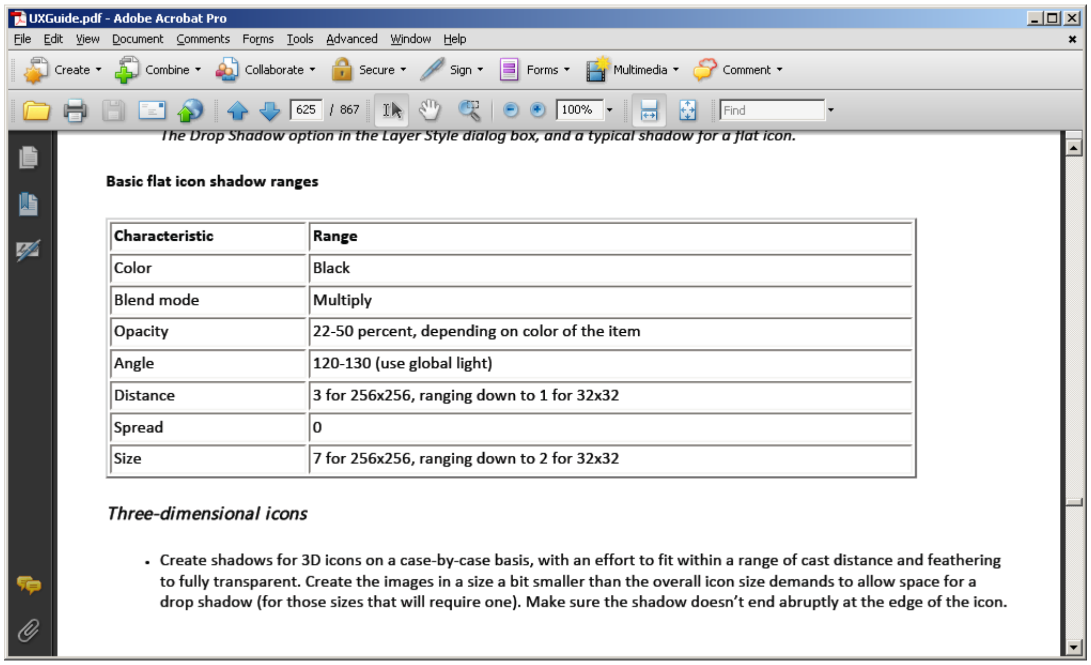
3.57.2.1 General

- Use shadows to lift objects visually from the background, and to make 3D objects appear grounded, rather than awkwardly floating in space.

- Use an opacity range of 30–50% for shadows. Sometimes a different level of shadow should be used, depending on the shape or color of an icon.
- Feather or shorten the shadow if necessary, to keep it from being cropped by the icon box size.
- Don’t use shadows in icons at 24 × 24 or smaller sizes.

3.57.3 Flat Icons

- Flat icons are generally used for file icons and flat real-world objects, such as a document or a piece of paper.
- Flat icon lighting comes from the upper-left at 130 degrees.
- Smaller icons (for example, 16 × 16 and 32 × 32) are simplified for readability. However, if they contain a reflection within the icon (often simplified), they may have a tight drop shadow. The drop shadow ranges in opacity from 30–50 percent.
- Layer effects can be used for flat icons, but should be compared with other flat icons. The shadows for objects will vary somewhat, according to what looks best and is most consistent within the size set and with the other icons in Windows Vista. On some occasions, it may even be necessary to modify the shadows. This will especially be true when objects are laid over others.
- A subtle range of colors may be used to achieve desired outcome. Shadows help objects sit in space. Color impacts the perceived weight of the shadow, and may distort the image if it is too heavy.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
625 / 867
100%
Find

The Drop Shadow option in the Layer Style dialog box, and a typical shadow for a flat icon.

Basic flat icon shadow ranges

Characteristic	Range
Color	Black
Blend mode	Multiply
Opacity	22-50 percent, depending on color of the item
Angle	120-130 (use global light)
Distance	3 for 256x256, ranging down to 1 for 32x32
Spread	0
Size	7 for 256x256, ranging down to 2 for 32x32

Three-dimensional icons

- Create shadows for 3D icons on a case-by-case basis, with an effort to fit within a range of cast distance and feathering to fully transparent. Create the images in a size a bit smaller than the overall icon size demands to allow space for a drop shadow (for those sizes that will require one). Make sure the shadow doesn’t end abruptly at the edge of the icon.

Figure 3-76: Basic flat icon shadow ranges

3.57.4 Color And Saturation

- Colors are generally less saturated than they were Windows XP.
- Use gradients to create a more realistic looking image.
- Although there is no specific color palette for standard icons, remember that they need to work well together in many contexts and themes. Prefer the standard set of colors; don't recolor standard icons, such as warning icons, because this disrupts users' ability to interpret meaning. For more guidelines, see Color.
- Icon files require 8-bit and 4-bit palette versions as well, to support the default setting in a remote desktop. These files can be created through a batch process, but they should be reviewed, as some will require retouching for better readability.
- Bit levels: ICO design for 32-bit (alpha included) + 8-bit + 4-bit (dithered down automatically—pixel poke only most critical). Only a 32-bit copy of the 256 × 256 pixel image should be included, and only the 256 × 256 pixel image should be compressed to keep the file size down. Several icon tools offer compression for Windows Vista.
- Bit levels: Toolbars 24-bit + alpha (1 bit mask), 8-bit and 4-bit.
- Toolbars or AVI files: Use magenta (R255 G0 B255) as the background transparency color.

3.57.5 Size Requirements

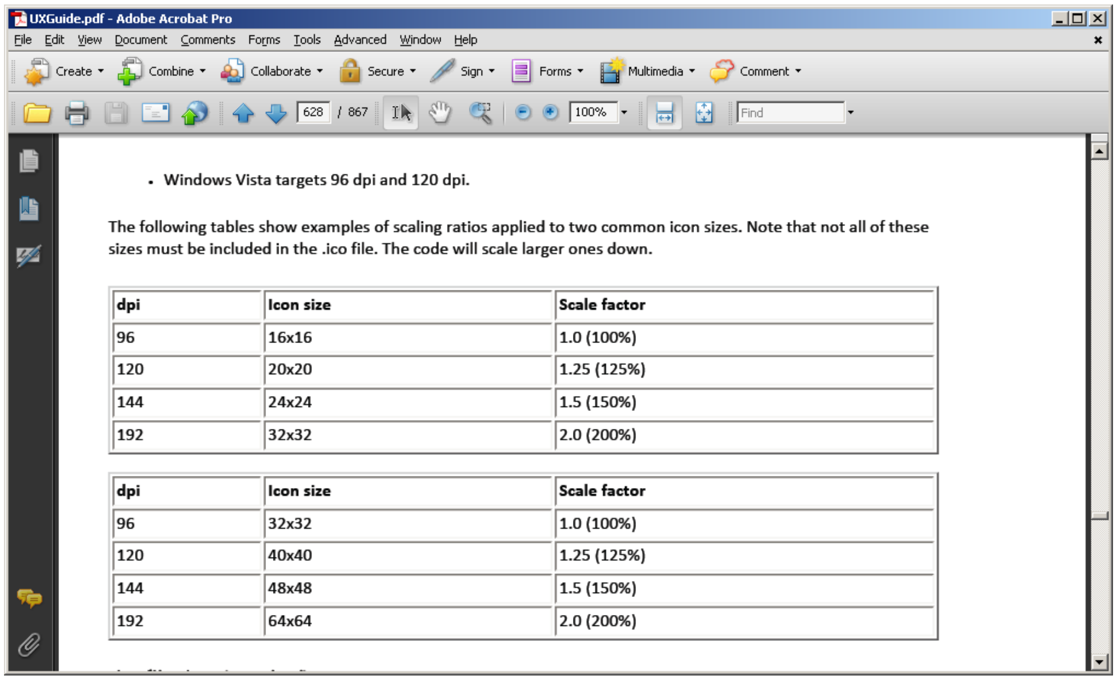
3.57.5.1 General

Pay special attention to high visibility icons, such as main application icons, file icons that can appear in Windows Explorer, and icons appearing in the Start Menu or on the desktop.

- **Application icons and Control Panel items:** The full set includes 16 × 16, 32 × 32, 48 × 48, and 256 × 256 (code scales between 32 and 256). The .ico file format is required. For Classic Mode, the full set is 16 × 16, 24 × 24, 32 × 32, 48 × 48 and 64 × 64.
- **List item icon options:** Use live thumbnails or file icons of the file type (for e×ample, .doc); full set.
- **Toolbar icons:** 16 × 16, 24 × 24, 32 × 32. Note that toolbar icons are always flat, not 3D, even at the 32 × 32 size.
- **Dialog and wizard icons:** 32 × 32 and 48 × 48.
- **Overlays:** Core shell code (for e×ample, a shortcut) 10 × 10 (for 16 × 16), 16 × 16 (for 32 × 32), 24 × 24 (for 48 × 48), 128 × 128 (for 256 × 256). Note that some of these are slightly smaller but are close to this size, depending on shape and optical balance.

- **Quick Launch area:** Icons will scale down from 48 × 48 in Alt+Tab dynamic overlays, but for a more crisp version, add a 40 × 40 to .ico file.
- **Balloon icons:** 32 × 32 and 40 × 40.
- **Additional sizes:** These are useful to have on hand as resources to make other files (for e×ample, annotations, toolbar strips, overlays, high dpi, and special cases): 128 × 128, 96 × 96, 64 × 64, 40 × 40, 24 × 24, 22 × 22, 14 × 14, 10 × 10, and 8 × 8. You can use .ico, .png, .bmp, or other file formats, depending on code in that area.

3.57.5.2 For High Dpi


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
628 / 867
100%
Find

- Windows Vista targets 96 dpi and 120 dpi.

The following tables show examples of scaling ratios applied to two common icon sizes. Note that not all of these sizes must be included in the .ico file. The code will scale larger ones down.

dpi	Icon size	Scale factor
96	16x16	1.0 (100%)
120	20x20	1.25 (125%)
144	24x24	1.5 (150%)
192	32x32	2.0 (200%)

dpi	Icon size	Scale factor
96	32x32	1.0 (100%)
120	40x40	1.25 (125%)
144	48x48	1.5 (150%)
192	64x64	2.0 (200%)

Figure 3-77: Windows Vista targets 96 dpi and 120 dpi

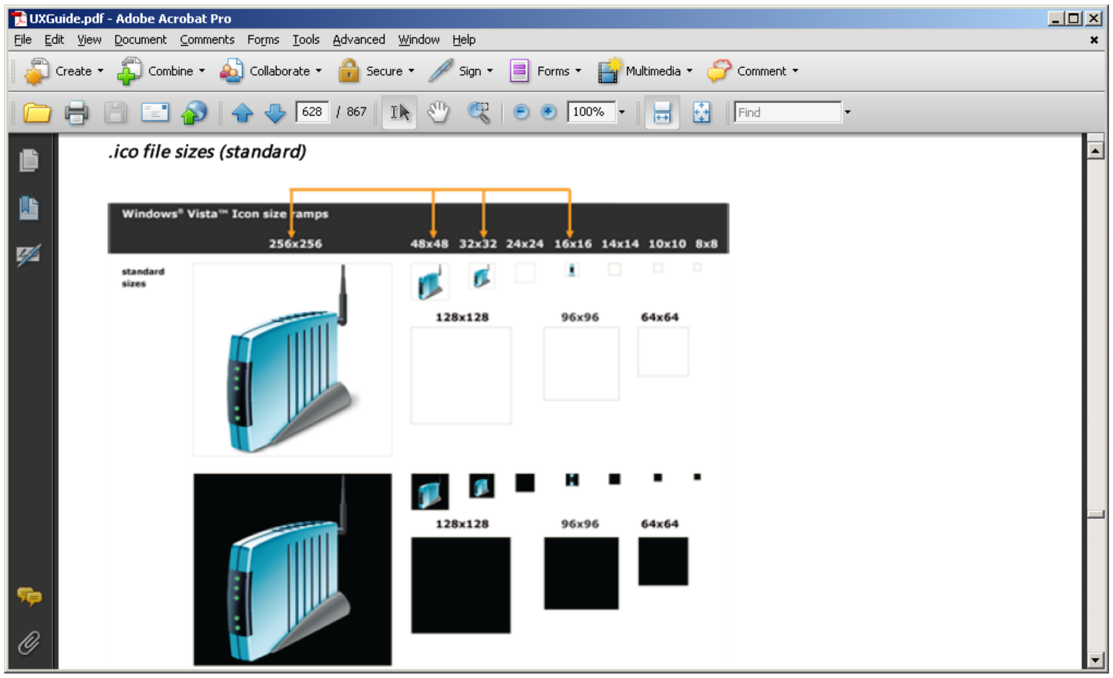

UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
628 / 867
100%
Find
.ico file sizes (standard)
Windows® Vista™ Icon size ramps
256x256
48x48 32x32 24x24 16x16 14x14 10x10 8x8
standard sizes
128x128
96x96
64x64

Figure 3-78: .ico file sizes (standard)

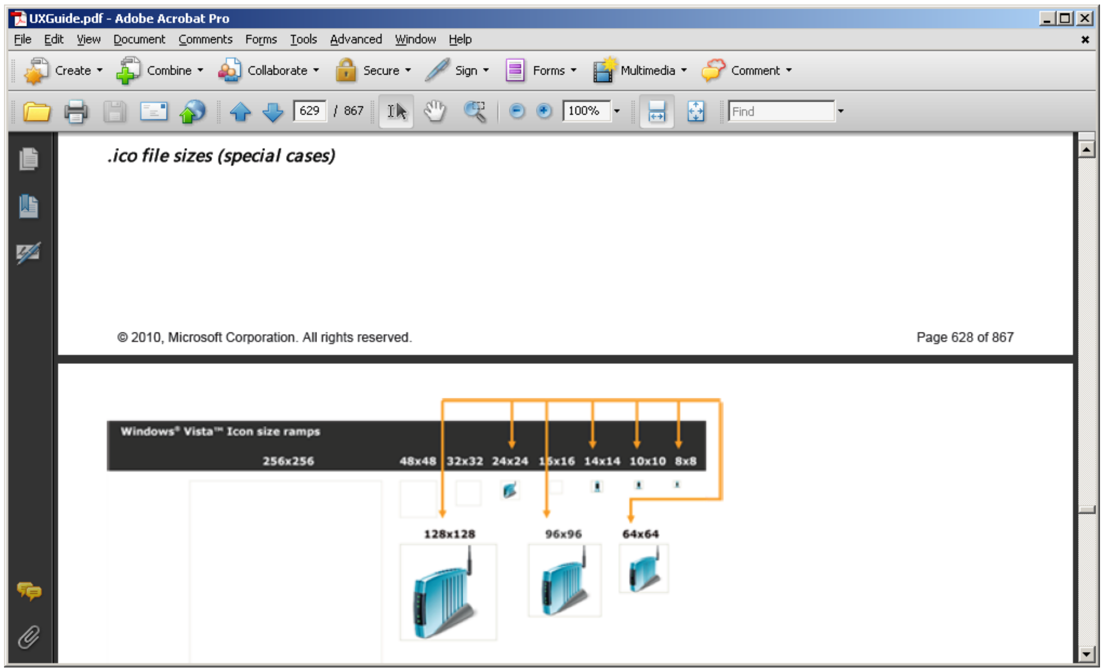

UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
629 / 867
100%
Find
.ico file sizes (special cases)
© 2010, Microsoft Corporation. All rights reserved.
Page 628 of 867
Windows® Vista™ Icon size ramps
256x256
48x48 32x32 24x24 16x16 14x14 10x10 8x8
128x128
96x96
64x64

Figure 3-79: .ico file sizes (special cases)

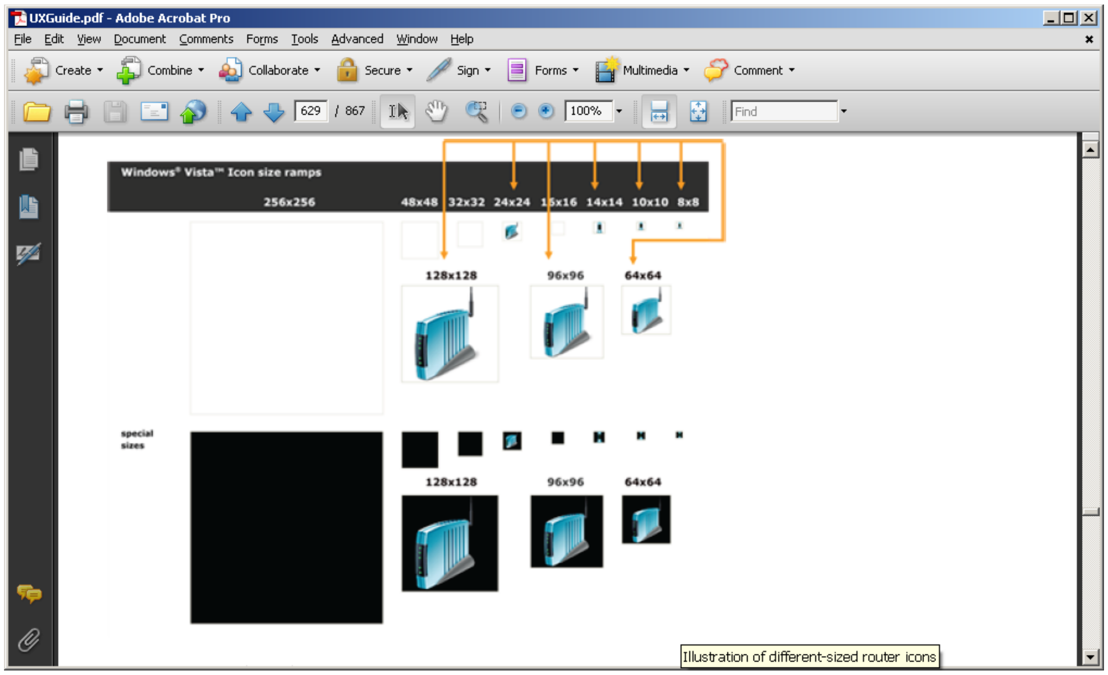

UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
629 / 867
100%
Find
Windows® Vista™ Icon size ramps
256x256
48x48 32x32 24x24 16x16 14x14 10x10 8x8
128x128
96x96
64x64
special sizes
128x128
96x96
64x64
Illustration of different-sized router icons

Figure 3-80: .ico file sizes (special cases)

3.57.6 Annotations And Overlays

- Annotations go in bottom-right corner of icon, and should fill 25% of icon area. **Exception:** 16 × 16 icons take 10 × 10 annotations.
- Don't use more than one annotation over an icon.
- Overlays go in bottom-left corner of icon, and should fill 25% of icon area. **Exception:** 16 × 16 icons take 10 × 10 overlays.

3.57.6.1 Level Of Detail

- 16 × 16 size of many of these icons is still widely used and therefore important.
- The details in an icon of this size must clearly show the key point of the icon.
- As an icon gets smaller, transparency and some special details found in larger sizes should be sacrificed in order to simplify and get the point across.
- Attributes and colors should be exaggerated and used to emphasize the key forms.
- Simply scaling down from the 256 × 256 size does not work.
- All sizes need relevant level of detail; the smaller the icon the more you need to exaggerate the defining details.

3.57.7 Icon Development

3.57.7.1 Designing And Producing Icons

- **Hire an experienced graphic designer.** For great graphics, images, and icons work with experts. Experience in illustrations using vector art or 3D programs is recommended.
- **Plan to do series of iterations,** from initial concept sketches, to in-context mock-ups, to final production review and fit-and-finish of icons in the working product.
- **Think ahead—icon creation can be expensive.** Gather all existing details and requirements, such as: the complete set of icons needed; the main function and meaning for each; families or clusters in the set you want to be apparent; brand requirements; the exact file names; image formats used in your code; and size requirements. Ensure up front that you can make the most of your time with the designer.
- **Remember that the designer may not be familiar with your product, so provide functional information, screen shots, and spec sections, as appropriate.**
- **Plan for geopolitical and legal reviews as appropriate.**
- **Map out a timeframe and have regular communication.**
- Create concept sketches.
- Try out the concept in different sizes.
- Render in 3D if necessary.
- Test sizes on different background colors.
- Evaluate icons in the context of the real UI.
- Produce final .ico file or other graphic resource formats.

3.57.7.2 Tools

- **Pencil and paper:** Initial concept ideas, listed and sketched.
- **3D Studio Max:** Render 3D objects in perspective.
- **Adobe Photoshop:** Sketch and iterate, mock-up in context, and finalize details.
- **Adobe Illustrator/ Macromedia Freehand:** Sketch and iterate, finalize details.
- **Gamani Gif Movie Gear:** Produce .ico file (with compression if needed).
- **Axialis Icon Workshop:** Produce .ico file (with compression if needed).
- Microsoft Visual Studio® doesn't support Windows Vista icons (there is no support for alpha channel or more than 256 colors).

3.57.7.3 Production

3.57.7.3.1 Step 1: Conceptualize

- Use established concepts where possible, to ensure consistency of meanings for the icon and its relevance to other uses.
- Consider how the icon will appear in the context of the UI, and how it might work as part of a set of icons.
- If revising an existing icon, consider whether complexity can be reduced.
- Consider the cultural impact of your graphics. Avoid using letters, words, hands, or faces in icons. Depict representations of people or users as generically as possible, if needed.
- If combining multiple objects into a single image in an icon, consider how the image will scale to smaller sizes. Use no more than three objects in an icon (two is preferred). For the 16 × 16 size, consider removing objects or simplifying the image to improve recognition.
- Do not use the Windows flag in icons.

3.57.7.3.2 Step 2: Illustrate

- To illustrate Windows Aero style icons, use a vector tool such as Macromedia Freehand or Adobe Illustrator. Use the palette and style characteristics as outlined earlier in this article.
- Illustrate image using Freehand or Illustrator. Copy and paste the vector images into Adobe Photoshop.
- Make and use a template layer in Photoshop to make sure that work is done within square regions of the regulated sizes.
- Create the images in a size a bit smaller than the overall icon size demands to allow space for a drop shadow (for those sizes that require one).
- Place images at the bottom of the squares, so that all icons in a directory are positioned consistently. Avoid cutting off shadows.
- If you are adding another object to an image or a series, keep the main object in a fixed position, and place flat smaller sized images in a fixed position, such as the lower-left or upper-right depending on the case.

3.57.7.3.3 Step 3: Create the 24-bit images

Once you've pasted sizes in Photoshop, check the readability of images, especially at 16x16 and smaller sizes. Pixel-poking using percentages of colors may be required. Reduction of transparency may also be needed. It is common to exaggerate aspects at smaller sizes and to eliminate aspects as well, in order to focus on the key point.

- The 8-bit icons will be displayed in any color mode lower than 32-bit and will not have the 8-bit alpha channel, so they may be hard to read).

- In Photoshop, duplicate the 24-bit image layer and rename the layer to 4-bit images. Index 4-bit images to the Windows 16 color palette.
- Clean up images using only the colors from the 16 color palette. Outlines made from darker or lighter versions of the object's colors are usually preferable to grey or black.
- If working on a bitmap, be sure that the background color isn't used in the image itself, because that color that will be the transparent color. Magenta (R255 G0 B255) is often used as the background transparency color.

3.57.7.3.4 *Step 4: Create the 8-bit and 4-bit images*

- Now that the 24-bit images are ready to be made into 32-bit icons, 8-bit versions need to be created.
- This is a great time to test contextual screen shots. It's amazing what can be discovered by viewing other icons or a family of icons in context. This step can save time and money. It is much better to catch issues before files go through production and are handed off.
- Add the drop shadow to your images in sizes that require them.
- Merge the drop shadow and the 24-bit images together.
- Create a new Photoshop file for each size. Copy and paste the appropriate image. Save each file as a .psd file.
- Do not merge the image layer with the background layer. It's helpful to include the size and color depth in the file name while working, but the file may ultimately need to be renamed.

3.57.7.3.5 *Step 5: Create the .ico file*

- Choose the application that best meets needs and skills of artists. Remember that icons to be used in a shipping product must be created in a tool that has been purchased or licensed. This means that trial versions cannot be used.
- Both of the products listed below have been used by designers who have produced icons for Windows Vista, and each offers the ability to export to Adobe Photoshop CS. Gamani Gif Movie Gear: Produce .ico file
- Axialis Icon Workshop: Produce .ico file
- Visual Studio doesn't support Windows Vista icons (there is no support for alpha channel or more than 256 colors), so its use is not recommended.
- Icon (.ico format) files must contain the 4- and 8-bit versions, as well as the 24-bit + alpha.
- Save files as a "Windows icon (.ico)" no matter which icon creation program you choose to use.

- Some iconographic assets may actually be bitmap strips, which also require an alpha channel (for example, for toolbars), or .png files saved with transparency. Not all are necessarily .ico format; check for what format is supported in code.

3.57.7.3.6 Step 6: Evaluate

- Look at all sizes.
- Look at the family together to evaluate family resemblance, optical balance, and distinction.
- Look at in context to evaluate relative weights and visibility (make sure that one doesn't dominate).
- Consider cases that may not be used now, but could be in the near future. Could this icon ever be annotated or have an overlay?
- Look at in code.

3.57.8 Icons In The Context Of List Views, Toolbars, And Tree Views

3.57.8.1 List Views

- For Windows Vista, use thumbnails for files holding content that is visually distinct at small scale, such that users can directly recognize the file they are looking for. (Use the Windows Thumbnailing application programming interface for this.)
- Application icon overlays (not shown here) on thumbnails help association with the application for the file type, in addition to showing the file's preview.

Note: For files without visually distinct content, don't use thumbnails. Instead, use traditional symbolic file icons showing object representation and the associated application or type.

3.57.8.2 Toolbars

- Icons that appear in a toolbar must have an optical balance in size, color, and complexity.
- Test potential icons in a contextual screen shot to avoid any undesired dominance or imbalances.
- Testing in screen shots easily helps avoid expensive iterations in code.
- Review the icons in code as well. Motion and other factors can impact the success of an icon; in some cases further iterations may be needed.

3.57.8.3 Tree Views

- Optical balance is needed to preserve the hierarchy in a tree view control.

- Therefore, icons that are typically used in this context should be evaluated there. Sometimes a particular 16 × 16 icon should be made smaller because its shape has an optical dominance over others.
- Compensation for optical imbalances is an important part of producing top quality icons.

3.58 Standard Icons (Page 635)

3.58.1 Guidelines

Note: For the following guidelines, “in-place” means on any normal window surface, such as within the content area of a wizard, property sheet, or control panel item page.

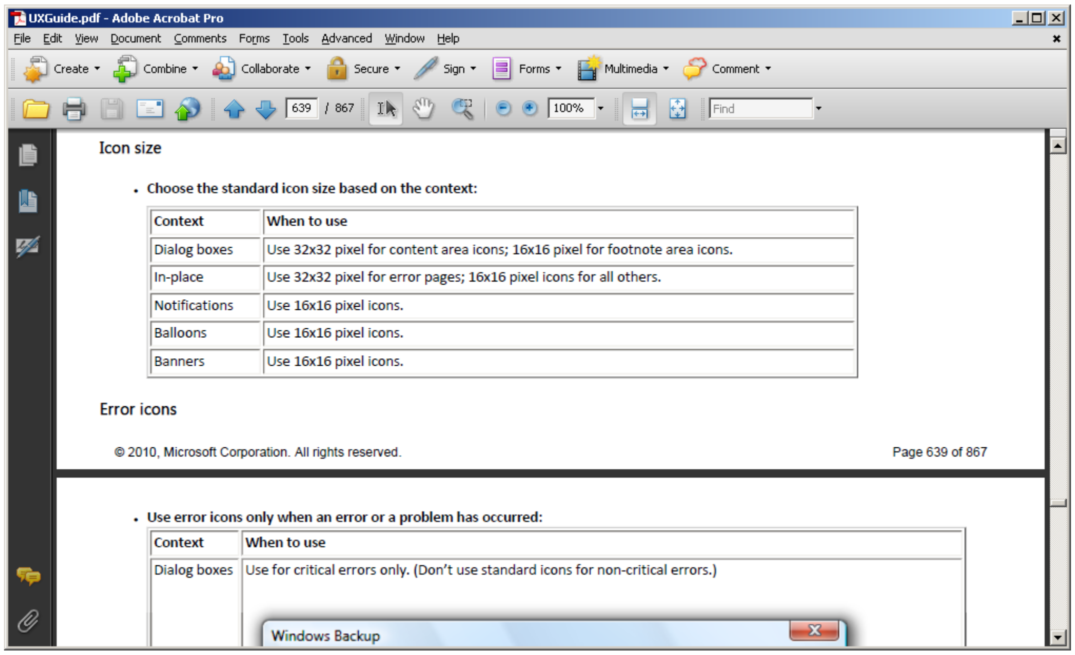
3.58.1.1 General

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 straddles 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, errors, because otherwise such contextual feedback would be too easy to overlook.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
639 / 867
100%
Find

Icon size

- Choose the standard icon size based on the context:

Context	When to use
Dialog boxes	Use 32x32 pixel for content area icons; 16x16 pixel for footnote area icons.
In-place	Use 32x32 pixel for error pages; 16x16 pixel icons for all others.
Notifications	Use 16x16 pixel icons.
Balloons	Use 16x16 pixel icons.
Banners	Use 16x16 pixel icons.

Error icons

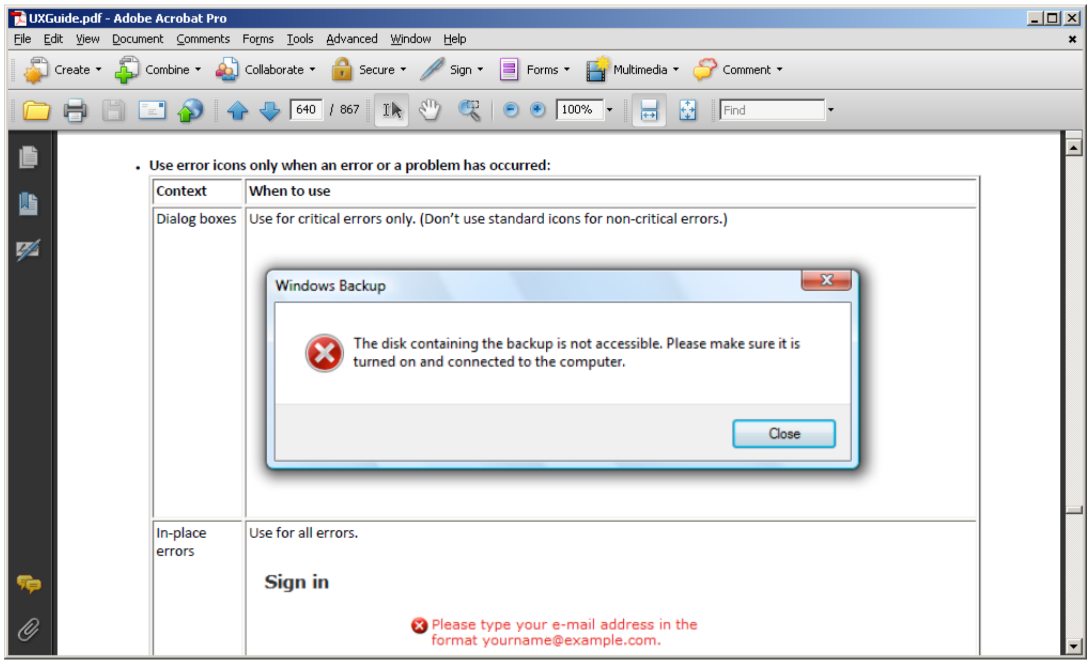
© 2010, Microsoft Corporation. All rights reserved. Page 639 of 867

- Use error icons only when an error or a problem has occurred:

Context	When to use
Dialog boxes	Use for critical errors only. (Don't use standard icons for non-critical errors.)

Windows Backup

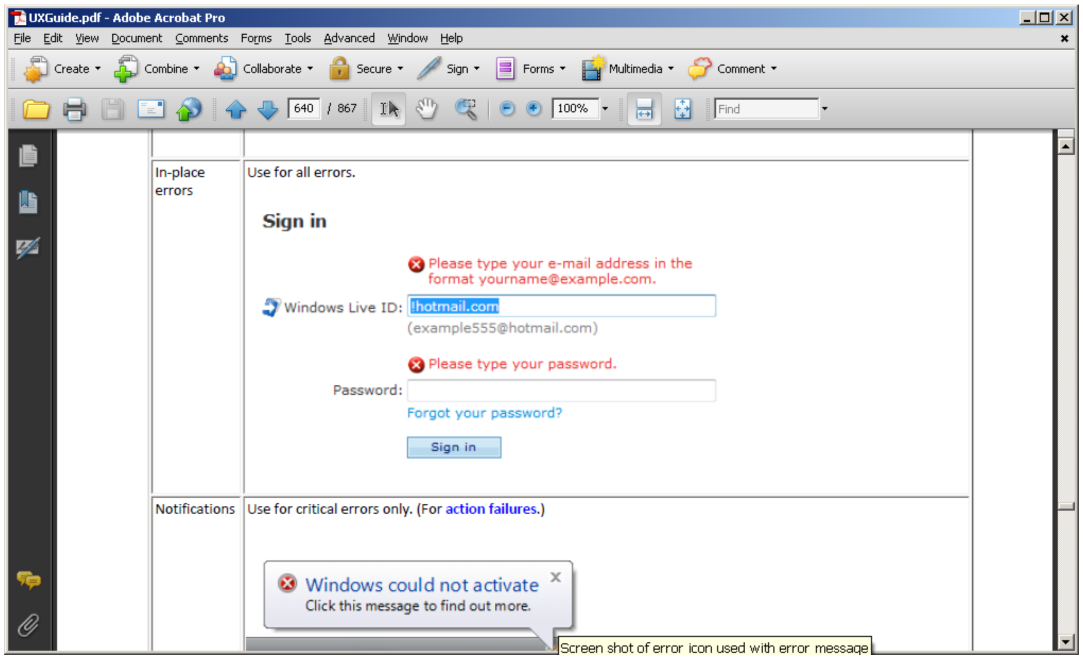
Figure 3-81: Icon size and error icons


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
640 / 867
100%
Find

- Use error icons only when an error or a problem has occurred:

Context	When to use
Dialog boxes	Use for critical errors only. (Don't use standard icons for non-critical errors.) Windows Backup: The disk containing the backup is not accessible. Please make sure it is turned on and connected to the computer. [Close]
In-place errors	Use for all errors. Sign in: Please type your e-mail address in the format yourname@example.com.

Figure 3-82: Windows backup error


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
640 / 867 100% Find

In-place errors	Use for all errors.
Notifications	Use for critical errors only. (For action failures.)

Sign in

Please type your e-mail address in the format yourname@example.com.

Windows Live ID: !hotmail.com

(example555@hotmail.com)

Please type your password.

Password:

Forgot your password?

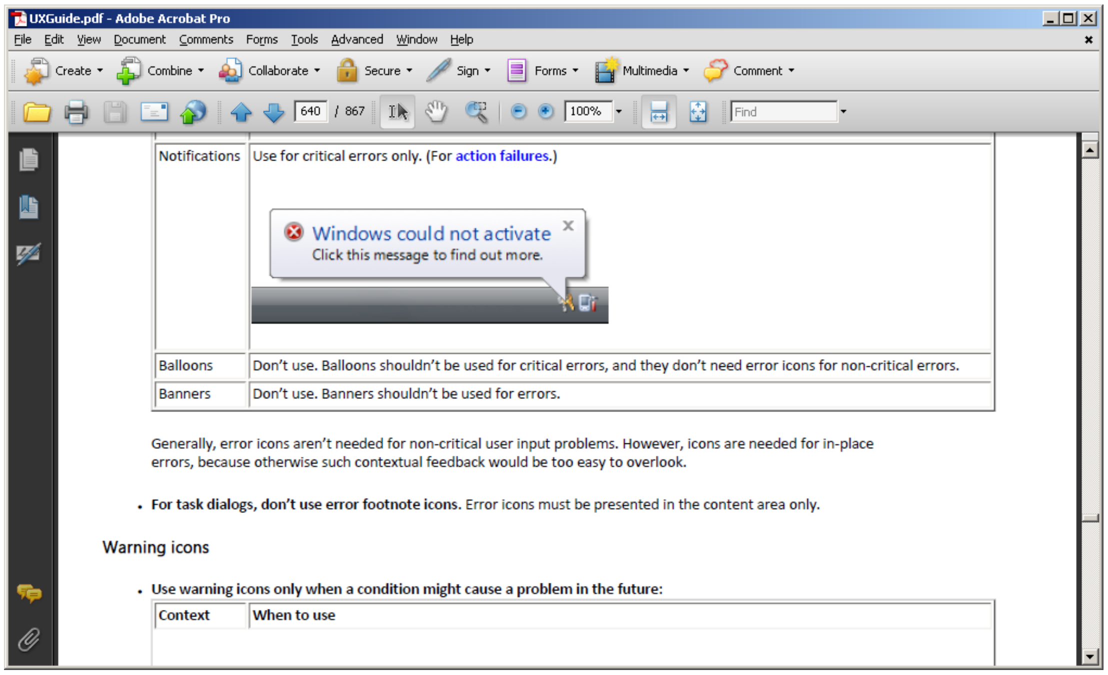
Sign in

Windows could not activate

Click this message to find out more.

Screen shot of error icon used with error message

Figure 3-83: Sign in errors


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
640 / 867 100% Find

Notifications	Use for critical errors only. (For action failures.)
Balloons	Don’t use. Balloons shouldn’t be used for critical errors, and they don’t need error icons for non-critical errors.
Banners	Don’t use. Banners shouldn’t be used for errors.

Windows could not activate

Click this message to find out more.

Generally, 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.

- **For task dialogs, don’t use error footnote icons.** Error icons must be presented in the content area only.

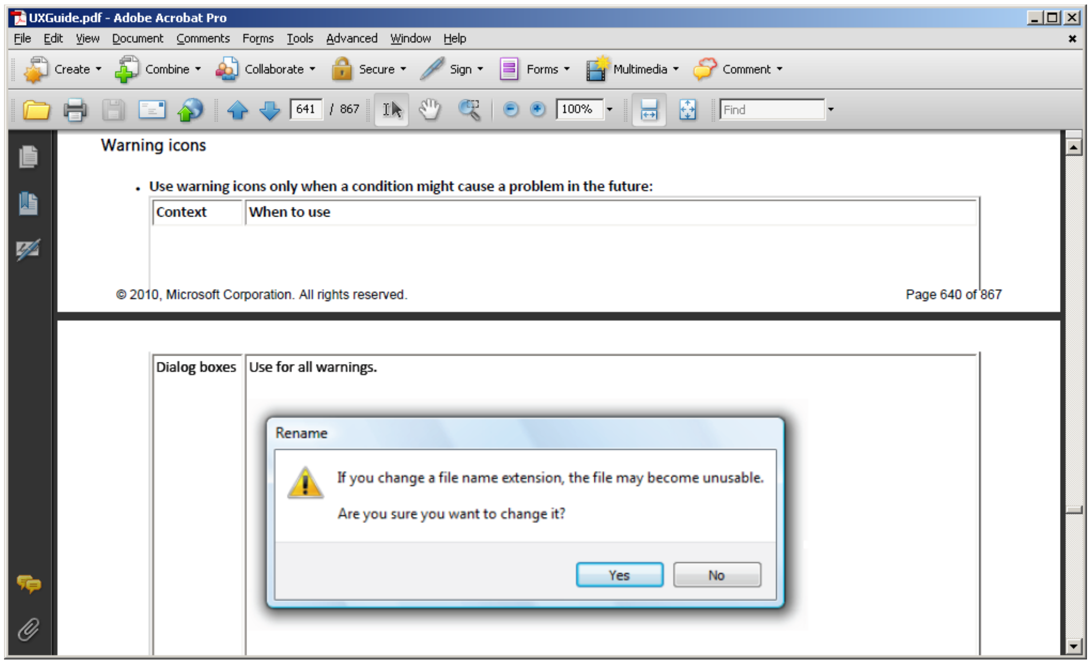
Warning icons

- **Use warning icons only when a condition might cause a problem in the future:**

Context	When to use

Figure 3-84: Windows could not activate error

- **For task dialogs, don’t use error footnote icons.** Error icons must be presented in the content area only.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
641 / 867
100%
Find

Warning icons

- Use warning icons only when a condition might cause a problem in the future:

Context	When to use
Dialog boxes	Use for all warnings.

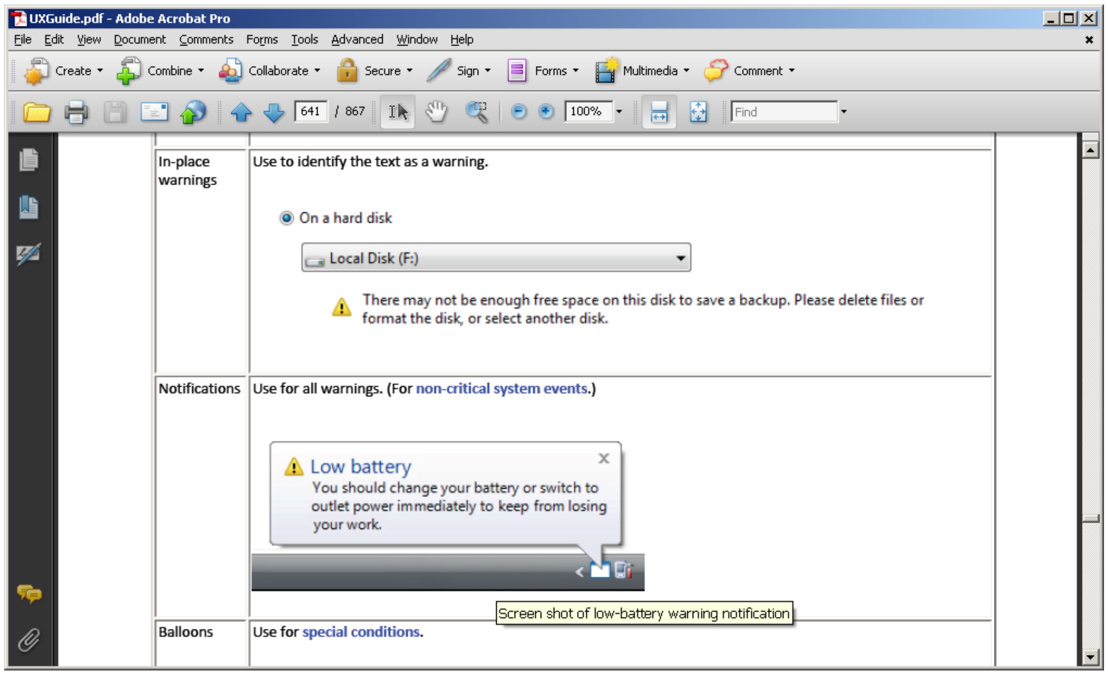
© 2010, Microsoft Corporation. All rights reserved. Page 640 of 867

Rename

If you change a file name extension, the file may become unusable.

Are you sure you want to change it?

Figure 3-85: Warning icons


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
641 / 867
100%
Find

In-place warnings	Use to identify the text as a warning.
Notifications	Use for all warnings. (For non-critical system events.)
Balloons	Use for special conditions.

On a hard disk

Local Disk (F:)

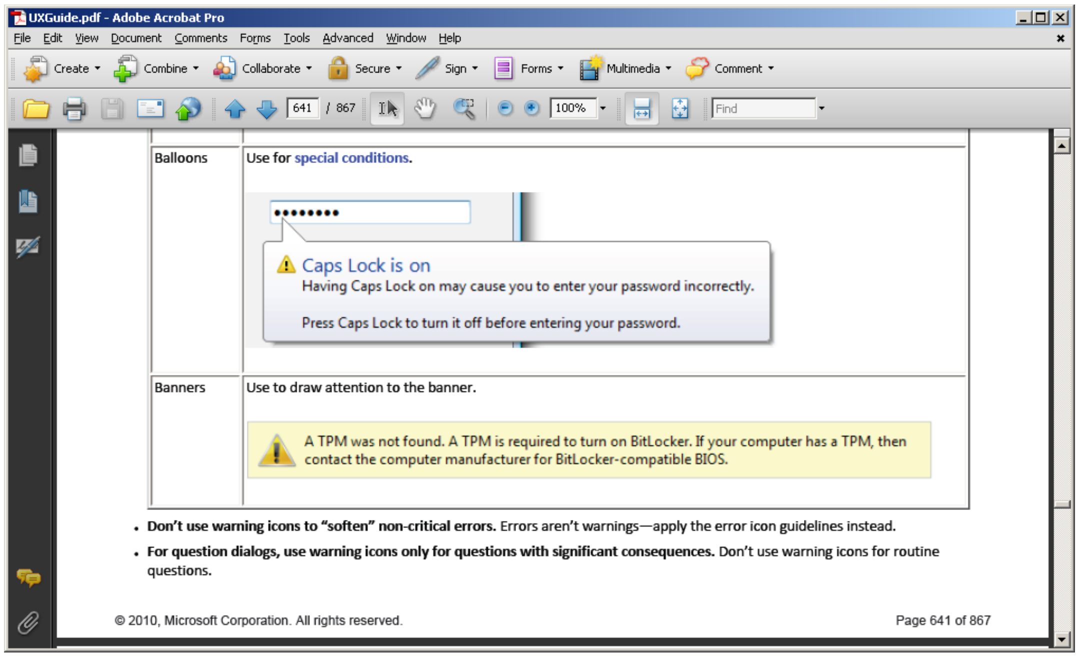
There may not be enough free space on this disk to save a backup. Please delete files or format the disk, or select another disk.

Low battery

You should change your battery or switch to outlet power immediately to keep from losing your work.

Screen shot of low-battery warning notification

Figure 3-86: Warning icons



UXGuide.pdf - Adobe Acrobat Pro

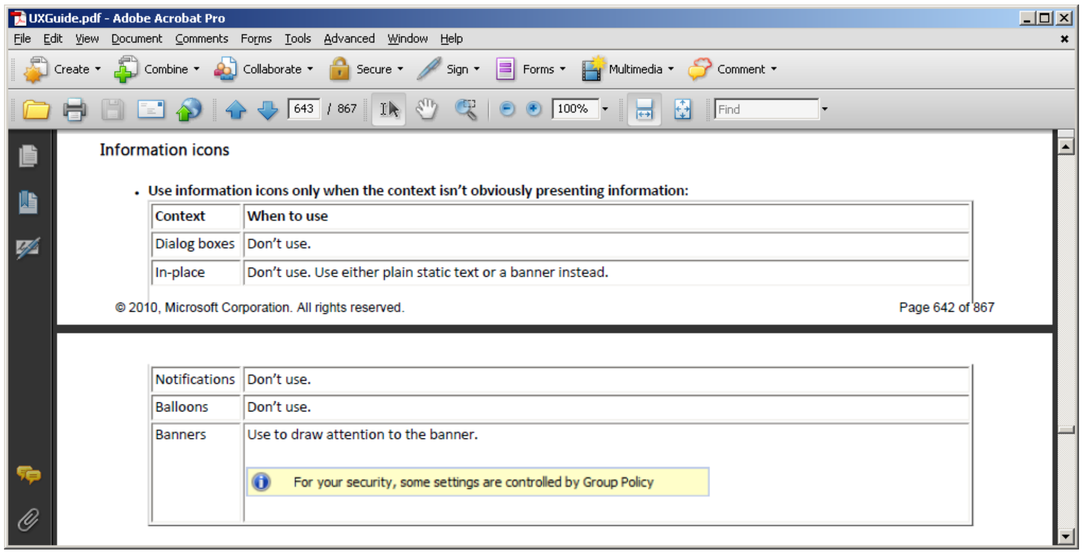
Balloons	Use for special conditions.
	Caps Lock is on Having Caps Lock on may cause you to enter your password incorrectly. Press Caps Lock to turn it off before entering your password.
Banners	Use to draw attention to the banner.
	A TPM was not found. A TPM is required to turn on BitLocker. If your computer has a TPM, then contact the computer manufacturer for BitLocker-compatible BIOS.

- **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.

© 2010, Microsoft Corporation. All rights reserved. Page 641 of 867

Figure 3-87: Warning icons

- **Don't use warning icons to "soften" noncritical 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.
- **For task dialogs, you can use a warning footnote icon to alert users of risky consequences.** However, use a warning icon either in the content area or the footnote area, but not both.



UXGuide.pdf - Adobe Acrobat Pro

Information icons

- **Use information icons only when the context isn't obviously presenting information:**

Context	When to use
Dialog boxes	Don't use.
In-place	Don't use. Use either plain static text or a banner instead.
Notifications	Don't use.
Balloons	Don't use.
Banners	Use to draw attention to the banner. For your security, some settings are controlled by Group Policy

© 2010, Microsoft Corporation. All rights reserved. Page 642 of 867

Figure 3-88: Information icons

3.58.1.2 Question Mark Icons

- Use the question mark icon only for Help entry points. For more information, see the Help entry point guidelines.
- Don't use the question mark icon to ask questions. Again, use the question mark icon only for Help entry points. There is no need to ask questions using the question mark icon anyway—it's sufficient to present a main instruction as a question.
- Don't routinely replace question mark icons with warning icons. Replace a question mark icon with a warning icon only if the question has significant consequences. Otherwise, use no icon.

3.59 Animations and Transitions (Page 660)

3.59.1 Guidelines

3.59.1.1 Effective Communication

- Define and use an animation vocabulary to ensure that your animations and transitions have a consistent meaning, and use it consistently throughout your program. Most vocabularies should include entries for scene and object appearance and disappearance, navigation, basic interaction (hovering, selecting, clicking), object manipulation and interaction (moving, dropping, resizing, scrolling, panning, zooming, rotating, filtering), and attracting attention. Consistent meaning is crucial to effective communication.
- Whenever practical, use the Windows animation vocabulary. While your program may have a different audience and different needs, often the benefits of consistency and familiarity outweigh the benefits of being different. If your program's vocabulary must be different, use the same basic animation types as Windows, but give them the right personality for your program.
- Don't assign specific meanings to generic animations and transitions in an animation vocabulary. Generic transitions like fades and special effects like dissolves have no particular meaning (beyond appear or disappear), so they can be used freely
- Make vocabulary entries clearly distinct. Related actions may have similar effects (for example, zooming in and zooming out should have inverse transitions), but unrelated actions should have clearly distinct effects (for example, zooming should never be confused with rotating).
- Keep real-world effects realistic and consistent. If you use realistic animations and transitions, keep the experience consistent with the real world. Users should never be surprised, confused, or mislead by the results. And for consistency, don't mix metaphors.

- Give inverse actions inverse animations. Doing so meets user expectations and simplifies the vocabulary. For example, if a pane appears by sliding in, remove it by sliding out—not with some other effect.
- Make animations comprehensible. Users should be able to understand quickly the purpose of an animation. It's possible to make an animation too small, too brief (less than 50 milliseconds), or so subtle that users aren't able to comprehend their purpose. In such cases, either redesign to make the meaning clear, or remove.

3.59.2 Patterns

3.59.2.1 Hover Feedback

- To appear responsive, strive to play animation within 50 milliseconds of entering or leaving the hover state.
- To appear fast, make the duration of hover animations less than 50 milliseconds.
- Use a fade in/fade out of hover effect. Doing so makes hover effects clearly distinct from click and selection feedback.

3.59.2.2 Click Feedback

- To appear responsive, strive to play animation within 50 milliseconds of click down event. Click up events don't need click feedback.
- To appear fast, make the duration of click animations less than 50 milliseconds.
- Use a background flash or blink effect. Doing so makes click effects clearly distinct from hover and selection feedback. Because clicking requires hovering, make click feedback a smooth addition to hover feedback.

3.59.2.3 Selection Feedback

- To appear responsive, strive to play animation within 50 milliseconds of selection or deselection.
- To appear fast, make the duration of selection animations less than 50 milliseconds.
- Use a fade in/fade out selection rectangle effect. Doing so makes selection clearly distinct from hover and click feedback.

3.59.2.4 Progress Feedback

- Use an activity indicator when an action can't be performed within a second. Doing so indicates that the command has been received.
- Use a progress bar when a task will take more than five seconds. For more guidelines, see Progress Bars.

- Use progress feedback animations that help users visualize the effect of long-running tasks. Avoid unnecessary progress feedback animations—if an animation doesn't communicate anything helpful, use a progress bar instead.
- Have clearly identifiable completion and failure states. Users must be able to determine these final states quickly.
- Stop showing progress when the underlying task isn't making progress. Users need to be able to determine if progress isn't being made, and react accordingly.

3.59.3 Attractors

- **Use attractors with restraint.** Unless the information is urgent, critical, or otherwise likely to affect the user's immediate behavior, it's usually better to change state inconspicuously and let users discover the change on their own. **Solve distractions, not discoverability.**
- **Choose an animation that draws the right level of attention.** Attractor animations should draw just enough attention to themselves to fulfill their purpose, but no more. If the user must act immediately, choose an effect that demands attention no matter where the user is looking. For other situations, refer to the **Attracting the right level of attention** section to get the right combination of attention, noticeability, and urgency.
- **If the user doesn't respond, don't repeat the animation or use continuous animations.** Instead, assume that the user chose not to act now, but may act later. Continuous animations make it difficult for users to concentrate on anything else.

3.59.4 Relationship Animations

- Use relationship animations to show where objects came from or where they went.
- Relationship animations must start or end with the selected object. Don't show relationships between objects the user isn't currently interacting with. If users notice at all, what they'll notice is the distraction.

3.59.5 Illustrations/Previews

- **Use previews to show the effect of a command without users having to perform it first.** By using helpful previews, you can improve the efficiency and ease of learning of your program, and reduce the need for trial-and-error.
- **Use illustrations and previews that have a clear interpretation.** They have little value if confusing.
- **Play only one illustration at a time** to avoid overwhelming users. If multiple simultaneous illustrations are possible, use mouse hover or a play button to let users indicate their interest.

- **Play an illustration automatically if it is the main purpose of the window or page.** Otherwise, if it's optional, let users play it when they are ready.
- **Play animations at the optimal speed**—not so fast they are difficult to understand, but not so slow they are tedious to watch.

3.59.5.1 Object Grow/Shrink

Don't clip content during a resize. Expand containers before adding content. Remove content before reducing containers.

3.59.5.2 Content Show/Hide/Change

Display important information statically. Users shouldn't have to access important information through progressive disclosure.

3.59.5.3 Control Or Affordance Show/Hide

- Display important controls when the user positions the pointer anywhere within the window or pane, or, if full screen, on mouse move. Users shouldn't have to hunt for these controls, so make their discovery certain.
- Display secondary controls or control affordances when the user positions the pointer on or near the commands. For easy discoverability, make the location obvious and the target large.

3.59.5.4 Scene Transitions

- **Make physical scene transitions consistent with natural mapping.** People read from left to right in Western cultures, and hierarchical diagrams flow from top to bottom. Consequently, going forward in time is indicated by left-to-right movement. The following physical scene transitions have natural mapping:
  – **TransitionMeaning** From leftMove back in task flowFrom rightMove forward in task flowFrom topMove up task hierarchyFrom bottomMove down task hierarchy
- **If your program plays sound, design scene transitions and audio transitions together.** For example, if a scene fades out gradually, any sound should fade gradually as well. Don't ruin seamless visual transitions by having abrupt sound transitions.

3.59.5.5 Direct Manipulations

- When using physical gestures in the interaction (like tossing), design the animation to feel like a natural response to the gesture. Link the interaction cause with the transition effect. Give the animation real-world physical characteristics such as acceleration, deceleration, momentum, resistance, weight, bounce, and rotation.

- To maintain a direct feel, keep an object's contact points under the pointer smoothly throughout the interaction. Any lag, choppy response, or loss of contact destroys the perception of direct manipulation. Objects should never disappear while being manipulated.

3.59.5.6 Sort, Filter, Or Reorder Transitions

- **For simple changes, show the entire transition.** Users will be able to follow the entire transition easily. Simple changes involve four items or fewer.
- **For complex changes, emphasize the end of the motion as it slows down, and let the eye fill in the rest.** Doing so makes the motion feel much more responsive and orderly.

3.59.5.7 Performance Transitions

- Consider performing slow transitions in two or three stages to make them appear faster and immediately interactive. Use the following composition order when appropriate:
  - External frame
  - Background
  - Initial content (using a temporary representation if necessary)
  - Primary controls (so that users can interact immediately)
  - Secondary controls and any remaining UI elements
  - Final content (if a temporary representation was used) Use transitions like fades and slides to make the composition appear smooth, orderly, and refined.

3.59.6 Special Experience Animations

- **Reconsider animated splash screens (as well as static splash screens).** Often splash screens just draw attention to how long a program takes to load, and they wear out their welcome quickly. While splash screens are acceptable if they are displayed only when user interaction isn't possible, whenever practical a better alternative is to design your program so that users can interact with it immediately, even while it is still loading.
- **Provide a Skip Introduction command if an animated splash screen takes more than three seconds.** Clicking anywhere on the splash screen should also dismiss it. Alternatively, use a short version of the animation after an initial period.

3.59.6.1 Performance

- **Don't make users wait for your program's animations and transitions.** Use brief animations and transitions (less than 200 milliseconds) whenever practical. Use faster animations (100 milliseconds) for more frequent operations.

- Design longer animations (more than one second—usually the progress feedback, illustration, and special experience patterns) so that users can continue to work while they are running.
- **Design long-running animations to make it clear to users that they can interact while the animation is running.** Users won't attempt to continue to work if the visual clues suggest that they can't.
  - **Use lightweight animations for CPU-intensive tasks.** Doing so gives full processing power to the task. Furthermore, users won't perceive that the lightweight animation is the reason why the task is CPU-intensive.
  - **Don't display an activity indicator during an animation or transition.** Doing so destroys the effect. Design animations and transitions so that they are able to start right away.
  - **Design animations to degrade gracefully whenever there are insufficient system resources.** Animations can degrade by having variations that require fewer resources (such as shorter lengths or lower frame rates), or even not running at all. Regardless of the resources available, make sure the animations have high quality and look like animations instead of software bugs.

3.59.7 Animation Characteristics

Well-designed animations and transitions generally have these characteristics:

- **Brief duration.** Most animations should be between 100 and 300 milliseconds, preferably either 1/6 second (167 milliseconds) or 1/4 second (250 milliseconds). (Special experiences and progress feedback can be longer.) Use faster animation times for more frequent operations. Generally, longer animations take more time to complete, take more time to understand, and feel slow.
- **Responsiveness.** Animations should start within 50 milliseconds of the initiating event or user action. Longer start times feel unresponsive.
- **Acceleration/deceleration.** To look natural, most animation effects need to accelerate when starting and decelerate when stopping. To look responsive, design animations to have fast starts. To appear controlled, design animations to have soft landings at the end. While this applies to motion effects, it also applies to any effect that suggests movement, such as zooms and even fades.

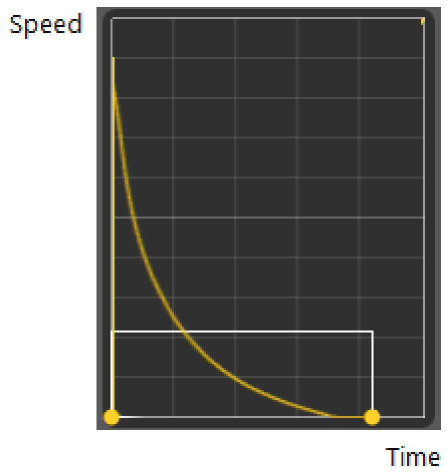


Figure 3-89: Speed vs. time

- **Motion.** Animations portraying motion in particular need to accelerate and decelerate, so don't use linear motion unless animation duration is very short. Motions should take the shorts path from beginning to end, without overshooting. The full motion path is not always required. When appropriate, emphasize the end of the motion as it slows down, and let the eye fill in the rest. Doing so makes the motion feel much more responsive and orderly. When animating the motion of several objects simultaneously, give them slightly different paths with slightly different timings to feel more natural.
- **Frame rate.** Most animations should use a frame rate of 20 frames per second. If the animation is for a special experience or is related to the main purpose of the program, consider using a higher rate of 24–30 frames per second to improve smoothness and realism.
- **Scale.** Design animations to work well across their entire range of intended usage. For example, page transitions should be designed to work for all page sizes.
- **Personality.** Design animations to feel natural, subdued, and efficient rather than artificial, whimsical, or slow.

3.59.8 Animated Text

- While you may display text using a transition, **don't continuously animate text.** Animated text is often distracting and more difficult to read than static text. **Exceptions:** You may animate text in situations where it is traditionally animated, and you provide an accessible alternative.
- You may animate text if the purpose of the text is primarily decorative

3.59.8.1 Reducing Power Consumption

- **Design your animations to reduce power consumption.** When designed properly, animations shouldn't increase power consumption significantly. To reduce power consumption: **Stop animating when the display is off.** The display may be off for the purpose of saving power.
- **Don't use long-running animations that aren't user initiated.** Animations that use high-resolution periodic timers reduce the efficiency of processor power management. Also, be sure to disable any high-resolution periodic timers when the animations are complete.
- **Suspend all animations when the system becomes idle.** The period of user inactivity to become idle is determined by Power Options in Control Panel.

3.59.8.2 Accessibility

- **Don't use animation as the only way to convey essential information.** Animations should communicate information that is useful but not critical, because they aren't accessible to users with visual impairments.
- **Make sure equivalent information is available through other means,** such as: **By inspection.** Users can determine equivalent information by looking at the screen or objects involved in the animation.
- **By simple interaction.** Users can determine equivalent information by hovering, clicking, or double clicking.
- **When appropriate, set input focus on the object changed during a transition.** Doing so enables assistive technologies to detect where the change happened. But don't change input focus when the user is using the keyboard.
- **Don't use animations or transitions that flash or resize objects quickly.** Flashing and rapid screen changes can cause problems for people with seizure impairments and other neurological disorders.
- **Allow users to turn off your program's animations and transitions.** To support this ability, respect the Turn off all unnecessary animations option in the Ease of Access Center in Windows.
- **Design tasks assuming that users will turn your program's animations off.** Make sure that doing so doesn't disrupt the task flow significantly.

3.60 Graphic Elements (Page 674)

3.60.1 Guidelines

3.60.1.1 General

Don't convey essential information through graphic elements alone. Doing so presents accessibility issues for users with visual disabilities or impairments.

3.60.2 Graphic Designs

- **Graphics are most effective when they reinforce a single simple idea.** Complex graphics that require thought to interpret don't work well. Hieroglyphics are best left for cave drawings.
- **Don't use arrows, chevrons, button frames, or other affordances associated with interactive controls.** Doing so invites users to interact with your graphics.
- **Avoid swaths of pure red, yellow, and green in your designs.** To avoid confusion, reserve these colors to communicate status. If you must use these colors for something other than status, use muted tones instead of pure colors.
- **Use culturally neutral designs.** What may have a certain meaning in one country, region, or culture may not have the same meaning in another.
- **Avoid using people, faces, gender, or body parts, as well as religious, political, and national symbols.** Such images may not easily translate or could be offensive.
- **When you must represent people or users, depict them generically;** avoid realistic depictions.

3.60.3 Backgrounds And Banners

- **To emphasize content, use dark text on a light background.** Black text on a light gray or yellow background works well.
- **To de-emphasize content, use light text on a dark background.** White text on a dark gray or blue background works well.
- **If a gradient is used, make sure that the text color has good contrast across the entire gradient.**
- **Always use a 16x16 pixel icon to draw attention to the banner.** Banners are too easy to overlook otherwise. For more guidelines and examples, see **Standard Icons**.
- **Use backgrounds and banners with caution.** While the intent of the background or banner may be to emphasize content, quite often the results are the opposite—a phenomenon known as "banner blindness."

3.60.4 Glass

- **Consider using glass strategically in small regions touching the window frame without text.** Doing so can give a program a simpler, lighter, more cohesive look by making the region appear to be part of the frame.
- **Don't use glass in situations where a plain window background would be more attractive or easier to use.**

3.60.5 Separators

- **Use vertical and horizontal lines for separators.** Be sure to have sufficient space between the separators and the content being separated.
- For separators between sizable content (splitters), display the resize pointer on hover.

3.60.6 Shadows

- Use only to make your program's most significant content or objects being dragged stand out visually against its background. Consider shadows to be visual clutter in other circumstances.

3.60.7 High dpi Support

- **Support 96 and 120 dots per inch (dpi) video modes.** Detect the dpi mode at startup and handle dpi change events. Windows is optimized for 96 and 120 dpi, and uses 96 dpi by default.
- **Prefer to provide separate bitmaps rendered specifically for 96 and 120 dpi over scaling graphics.** At least provide 96 and 120 dpi versions for the most important, visible bitmaps, and either center or scale the others. Such applications are considered "high-dpi aware" and provide a better overall visual experience than programs that are automatically scaled by Windows. Developers: You can declare a program high-dpi aware (and prevent automatic scaling) setting the dpi aware flag in the program's manifest, or by calling the SetProcessDPIAware() API during program initialization. You can use macros to simplify selecting the right graphics. For Win32 bitmaps, you can use SS_CENTERIMAGE to center or SS_REALSIZECONTROL to scale.
- Check your program in both 96 and 120 dpi for:
  - Graphics that are too small or too large.
  - Graphics being clipped, overlapped, or otherwise not fitting properly.
  - Graphics that are poorly stretched ("pixilated").
  - Text that is clipped or not fitting in graphic backgrounds.

3.60.8 Text

For accessibility and localization, don’t use any text in graphics. Make exceptions only to represent branding and text as an abstract concept.

3.61 Sound (Page 677)

3.61.1 Guidelines

3.61.1.1 Usage

- **Use sound with restraint**—make sure there is a clear overall user benefit. Focus on sounds that keep users informed, are likely to change their behavior, or provide useful feedback. When in doubt, don’t use sound.
- **Select the sound and its characteristics based on how it is being used.** For a description of each usage pattern, see the table in the previous section.
- **For notifications and feedback, don’t use sound as the only method of communication.** Rather, use sound as a supplemental method to reinforce visual or textual cues. Doing so ensures that users can see the information if they can’t hear the sound.
- **Don’t play loud or harsh sounds frequently.** Doing so is unnecessary and results in a poor user experience. The more often a sound is played, the less obtrusive it should be. Sounds don’t have to be loud or harsh to attract attention.
  - **Don’t beep.** Beeping isn’t appropriate for modern programs. Beeps can’t have specific meanings assigned to them, and users can’t control them. **Exception:** Critical system functions may beep to alert users of situations that they must attend to immediately, such as critically low battery power.

3.61.1.2 Playback

- Don’t repeat a sound more than two times consecutively.
- For a consecutive sequence of related sound events, play a sound only on the first event. Avoid using multiple sounds because they may collide with each other or otherwise result in an unpleasant user experience.

3.61.2 Sound Selection

- **Choose pleasant sounds.** Don’t use unpleasant, alarming sounds, such as buzzing, crashing, and breaking.
- **Use sounds that are short** (less than one second).
- **Use sounds that are roughly the same volume as the typical Windows sound.** Users dislike having to turn the volume down when starting a computer or a program, especially in public environments such as meetings and presentations.

- The Microsoft® Windows® sound files are located in the Media folder within the Windows folder.
- **Don't choose sounds that require localization.** You can achieve this by using sounds that don't use speech or have culturally-dependent meanings or connotations.

3.61.3 Windows System Sounds

- **Use the built-in Windows system sounds whenever appropriate.**
- **Choose to use system sounds based on their associated meaning, not just on the sound itself.** System sounds must be used consistently.

3.61.4 Sound Design

When creating your own sounds:

- **Create sounds with the desirable sound characteristics.**
- **Compose sounds with mostly mid-range to high frequencies (600 Hz to 2 kHz).** Don't use low frequencies because they travel farther, are harder to locate, and can be alarming.
- **Set the relative amplitude of normal sounds to the level of the typical Windows sound.** The Windows sounds have been appropriately leveled for home and work environments. Using different levels for your sounds will force users to make volume adjustments. Set important sounds to be slightly louder. Such sounds include action completions, action failures, and important system events.
- Set frequently occurring sounds to be slightly softer. These include FYIs, branding sounds, and sound effects.
- **Choose sounds consistent with the meaning of the Windows sounds.** To create a custom version of a Windows sound, preserve the same pitch and interval, but change the orchestration or timbre. Don't assign different meanings to sounds with similar pitches and intervals as Windows sounds.
- **Design the sounds for your program to feel like they are related variations on a theme.** Your program's auditory experience should be coordinated with its visual experience. **Design scene transitions and audio transitions together.** For example, if a scene fades out gradually, any sound should fade gradually as well. Don't ruin seamless visual transitions by having abrupt sound transitions.
- **Sounds must be in .wav file format.** The 16-bit, 44.1 kHz stereo uncompressed pulse code modulation (PCM) format is recommended. If file size is important, use compressed or monaural (mono) formats, but be aware that there is an easily discernable quality loss that could reflect badly on your application.

3.61.5 Mixing

- **Don't have volume or mute controls in your program.** Instead, let users control relative volume settings among applications with the Windows volume mixer. If your program has a volume control, there will be multiple places where users adjust their settings, which may lead to confusion.
- **Exception:** If the primary purpose if your program is audio or video playback or creation, it may be useful to have a volume control in the program. Use a slider control for this purpose and provide immediate feedback when the user changes the volume.

3.61.6 Windows Integration

- **Register your program's sounds in the Windows Sounds registry.** Doing so allows the Windows volume mixer to add a slider for your program.
- **Register your program's custom sound events.** Doing so allows the Windows Sound control panel item to display them. Create the following key for each custom sound event: HKEY_CURRENT_USER | AppEvents | Event Labels | EventName = *Event Name*.
  - **Don't hardwire the sounds for your program's sound events.** Instead, specify the sounds to be played using registry entries. Doing so allows users to customize the sound events through the Sound control panel item. **Exception:** You can choose to hardwire sounds used for branding.
  - **Don't provide a way for users to configure sounds within your program's options.** Rather, use the Windows Sounds control panel item for this purpose.
  - **Consider not assigning sounds to frequently occurring events by default.** Don't require users to configure their way out of an annoying initial experience.

3.61.7 DirectSound Programming Issues

- For DirectSound programs that have their own volume control, **set the program volume to 100% by default.** Doing so maximizes the dynamic range of your audio.
- **Don't lock out other sound events by running your program in exclusive mode.** Doing so can prevent other programs from working correctly. For example, using exclusive mode prevents a computer from being used as a telephony device.

3.61.8 Text

- Don't use the phrase "sound adapter". Use "sound card" instead.
- Use "device" to refer generically to speakers, headphones, and microphones.

- Use "controller" to refer to audio hardware that controls devices, such as sound cards and chipsets.
- Use the phrase "sound scheme" to describe a collection of sounds for common program events, such as logging on or receiving new e-mail. Use the phrase "desktop theme" to describe a collection of visual elements and sounds for your computer desktop.
- Use the term "audio" to refer broadly to speech, music, and sounds. Use the term "sound" to refer more narrowly to the program and Windows sounds described in this article.

3.62 Experiences (Page 686)

3.62.1 Guidelines

3.62.1.1 General

- Choose a good product name that is memorable, distinctive, and concisely conveys the benefit of the product. This will be the foundation of your brand.
- Focus your branding effort on the special experiences in your program, such as:
  - The **first experiences**, especially during setup and when the program is used for the first time.
  - The main window or home page.
  - The start and completion of important tasks.
  - Important transitions between tasks or program areas.
  - Log in and log off.
- **Prefer secondary branding elements.** Limit your use of primary branding elements to a few strategic experiences. For example, consider using secondary graphics, transitions, and color instead of logos. Also, avoid prominent primary branding elements in places where users spend a lot of time because they may be perceived as clutter.
- **Don't use branding that is distracting or harms usability or performance.**
- **Don't use the Windows desktop or Start menu for branding.**

3.62.2 Names And Logos

- **Limit the use of product and company logos in the user interface.** Don't plaster company or product logos on every UI surface. Limit product and company logos to at most two different surfaces, such as the main window or home page and the About box.
- Limit product and company logos to at most twice on any single surface.

- Limit product and company names to at most three times on any surface.
- **Use small product and company logos.** Place the logo out of the user's workflow, and choose a size that is appropriate for its location.
- **Use graphic logos.** Graphic logos are more stable than text logos because they aren't affected by font, text size, language pack, or theme changes.
- **Don't use animated logos.**

3.62.3 Controls

Don't use custom controls for branding. Rather, use custom controls when necessary to create a special immersive experience or when special functionality is needed.

3.62.4 Splash Screens

- **Don't use splash screens for branding.** Avoid using splash screens because they may cause users to associate your program with poor performance. Use them only to give feedback and reduce the perception of time for programs that have unusually long load times.
- **Don't use animated splash screens.** Users often assume that the animated splash screen is the reason for a long load time. Too often, that assumption is correct.

3.62.5 Sound

- Generally, sound is not recommended just for branding. If you do use sound for branding: Play a sound only at program startup, but only if the program was launched by the user.
- Synchronize the sound to a visual event, such as a UI transition like the display of a program window.

3.63 Setup (Page 693)

3.63.1 Guidelines

3.63.1.1 General

- **Apply the standard wizard guidelines for wizard-based setup programs.** Use these guidelines to determine good page design, effective navigation, good control labels, use of main instructions, and use of help.
- **Allow users to restart the setup program where they left off if it requires a lot of user input or takes a long time to complete.** If users restart the program after closing it before completion, restore previous user input, and restart where the setup was stopped.

- **Don't display setup windows maximized.** Displaying a setup window maximized presumes that users will give setup their undivided attention, which is unlikely. Instead, choose a size that is appropriate for the content to maintain a simple appearance.

3.63.2 Windows Integration

- **Name the setup file "Setup.exe".** "Install.exe" is an acceptable alternative. This enables Windows (and users) to recognize the file as a setup program. **Exception:** For programs downloaded from the Internet, help users manage and organize their Downloads folder by including the name of the program in the name of the setup file. For example, SetupVisualStudioExpress2008.exe.
- **Copy program files to the proper file system locations.** Doing so allows users and Windows to find and organize the files better.

3.63.3 User Account Control

- Digitally sign the setup executable file. Signed executables have many advantages, including using a more specific User Account Control Elevation UI. For information about signing files, see **Introduction to Code Signing**.
- **If a setup might require elevation, elevate as late as possible.** Display the elevation UI only after the user has committed to an option that requires elevation. Usually, the elevation UI appears during the installation phase, not the decision phase. However, if a setup always requires elevation, elevate at its entry point.
- **Always require elevation for uninstall.** Doing so prevents malware from uninstalling critical software without users knowing about it.
- **Once elevated, stay elevated until elevated privileges are no longer necessary.** Users shouldn't have to elevate multiple times to perform complete a program installation.
- **If special privileges are required for installation, verify the user's credentials and report any problems on the first or second page.** Don't let users perform a lot of work only to find that they don't have the right credentials to complete the installation.
- **Require the least privileges possible.** For example, administrators are reluctant to install software that requires domain administrator credentials.

3.63.4 Restarting Windows

- **Avoid restarting Windows.** Most programs should install without restarting Windows. The primary reason program installations or updates require a system restart is that some of the files involved are currently being used by a running program. In this case, a better alternative is to make users aware of the situation, allow users to close these programs, and retry the action.

  - **If your setup must restart Windows: Use a single restart.** Delay the restart required by any prerequisites until the program and its updates are completely installed.
  - **Let users determine when it happens.** Don't restart Windows automatically, because users may lose work. Make sure that it's clear to users that they have a choice.
  - **If the user chooses to not restart Windows immediately, present any final feedback as a success, not a failure.** While technically the installation isn't complete until restart, it was successful from the user's point of view. See **Error pages** for an example.

3.63.5 Streamlining Setup

- **Whenever practical, start the installation process with a single step.** For example, instead of adding a separate page in setup for the license terms, you may provide a link to them instead. If you link to the terms: Phrase the commit button as "Agree and install" to require explicit consent to accept the license terms.
- Ensure that the license agreement link cannot be broken by linking to a file local to the setup instead of a Web page.
- Provide the ability to print the license agreement from its display window.
- **Eliminate unnecessary options and questions.** Postpone options that are more appropriate for the first use of the program or feature.
  - Don't ask users questions about the system state. Detect this information automatically instead, and ask users to verify only if there is a reason to change.
  - Don't ask questions about unimportant details. For example, for typical Windows programs it is safe to assume that you should copy program files to the Program Files folder.
  - Don't ask permission to do what you shouldn't do anyway. For example, most programs shouldn't include an option to put the program icon on the desktop.
  - Don't confirm setup cancellation. If users click Cancel during setup, assume the cancellation was intentional and close the program without confirmation. If doing so risks losing significant time or effort, allow users to restart your setup program and pick up where they left off.
- **Optimize for unattended installation.** Present all options and questions during the decision phase.
- For the download and installation phases, delay requiring user input to any problems encountered until the end of the phase. By doing so, users can leave the installation unattended until they return at their convenience.
- **Eliminate unnecessary pages.** If most users always just click Next on a page, consider getting rid of the page.

- **Eliminate unnecessary text.** Remove redundant text from instructions and labels.
- Don't explain basic Windows usage concepts
  - How wizards work
  - How setup works
- **Eliminate unnecessary effort.** Provide good default values:
  - Generally, select the most secure and private response to be the default.
  - If safety and privacy aren't factors, select the most likely or convenient response.
  - If an option is strongly recommended, consider selecting it by default, or adding "(recommended)" to its label.
- Advance pages automatically when a page has no input and the task is done successfully, such as with download, installation, progress, and updates pages. Once the step is done, stay on these pages only to show problems.
- When practical, start the program automatically when setup is done, instead of showing a Congratulations or Completion page. When setup is run interactively, assume the user is installing your program in order to run it immediately, so running the program is the best feedback to show that setup is complete. Automatically running the program isn't practical when the setup installs more than one program (for example, a suite consisting of many programs), when setup isn't run interactively, or when the installation process isn't complete after setup.

3.63.6 Page Types

3.63.6.1 Welcome and Getting Started Pages

- **Eliminate Welcome pages.** While it's great to feel welcome, users typically just click Next without reading. And because users typically skip over these pages without reading, the text does little more than state the obvious, by design.
- **Use a Getting Started page only if you must inform users about prerequisites for installing.** Such prerequisites include installing required software or hardware, performing required system configuration changes and updates, performing a system backup to protect against data loss, or obtaining required information that the user isn't likely to have already.
- **Whenever practical, provide the ability to perform the prerequisites directly from the setup program.** Users should have to perform the steps manually only if there isn't an alternative.
- If a Welcome page or Getting Started page isn't used, **include the program name and description on whatever is the first page of the Setup program.** You can use welcoming language as introductory text as long as the page's purpose is clear.

3.63.6.2 License Terms Pages

- Write the license terms using clear, concise text. Use plain language. Avoid "legalese."
- Present using a format that is easy to read and scan. Don't use long passages of uppercase text.
- Require explicit consent to accept the license terms. License acceptance should never be selected by default. If radio buttons are used to indicate acceptance, leave the options cleared by default and require users to accept the terms before enabling the Next button.
- Don't require users to scroll to the bottom of the license terms text before the Next button is enabled. This imposes an unnecessary burden on users to understand why the Next button is disabled.
- Provide a Print command, either with a command button or a context menu. Present the terms in a format optimized for printing.

3.63.6.3 Product Registration Pages

- Require users to register only if they must in order to use the program. Clearly explain why users must register.
- Provide optional registration only if there is a clear user benefit, such as to notify users of product updates. Leave this option cleared by default.
- Allow users to register later. Provide a maximum of three reminders and allow users to dismiss the reminders with a single click.

3.63.6.4 Scope Pages (Typical, Custom, Or Minimum)

- **Prefer to eliminate this page.** Assume that most users want the typical setup experience (and design that experience so that it works well for most users).
  - If you must include a scope page: **Explain the differences among options in terms of functionality and disk space.** Users rely on the clarity of information on the scope page to ensure that they make the right choice.
  - **Make sure that the custom options are necessary only for a small percentage of users, while most users can safely ignore them.** If not, the options should be in the typical setup path.
  - **If users choose custom options, have the typical installation options selected by default.** Users regard the typical installation as the baseline, and want to customize by adding or removing options from that baseline.
  - If you must use a custom installation option, **consider using relative button sizing and placement to guide most users to the typical installation.**

3.63.6.5 Input Pages

- Reduce the number of setup options by doing the right thing by default. For ways to eliminate options see Streamlining setup.
- Provide acceptable default values whenever possible. Choose defaults that are secure and private, and are acceptable to most users without change.
- Unless your program has unusual requirements, strive to have a single page of questions and options. But if your program requires several pages of questions and options, display them in the main wizard page flow. Don’t try to reduce the number of pages technically by putting options in dialog boxes or using tabs.
  - **Validate input as soon as possible:** Prohibit invalid characters on entry.
  - Use **balloons** to report problems with invalid text boxes.
  - Validate related fields on a page when users click Next.
  - Validate related fields across input pages as soon as problems can be detected.
- **Give all editable file paths a Browse button.** Allow users to specify network paths.
- **For the final input page, label the commit button Install, not Next.** Users shouldn’t be surprised by when the installation starts. Before the commit point, make sure that users can easily change any settings.

3.63.6.6 Start Installation Pages

- **Eliminate this page if it has no purpose other than to summarize the previous choices and begin installation.** If the input pages are clear and few in number, there should be no need to summarize them. Instead, the final input page should have the Install button, leading directly to the progress page.
- **For complex installations targeted at IT professionals, provide an Installation page with a comprehensive list of changes that the setup program will perform.** Many IT professionals have strict change management control, so they need to know the effect installing the program will have in detail.

3.63.6.7 Progress Pages

- **Always provide a progress page,** even if the program installs quickly. Provide a separate progress page for **the downloading phase** if there is one. Disable the Back (or Previous) and Next buttons while the setup is in progress, but leave the Cancel button enabled and responsive.
- **Use a single, determinate progress bar.** Follow the **determinate progress bar guidelines**, including: Clearly indicate completion. Don’t let a progress bar go to 100 percent unless the operation has completed.

- 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 a concise description of the current step above the progress bar.** For quick installations, such text is unnecessary; the progress bar alone is sufficient. For installations requiring a minute or longer, text can be helpful for users attending the setup. **Use sentence fragments, typically beginning with a verb, and ending with an ellipsis.**
- **Place text above the bar, not below.**
- **Refrain from cluttering the progress page with unnecessary details.** This page isn't for **technical support**, so there's no need to display registering GUIDs or specific files copied.

3.63.6.8 Error Pages

- If setup fails with a significant problem, display an error page that explains the problems along with practical steps to resolve them. Display the page with an error icon. Don't use a dialog box for this purpose.
- If setup completes with a minor recoverable problem, present the problem as an additional task instead of an error. Use positive, success-oriented, encouraging language, not terms like *error, failure*, or *problem*. Don't use an error icon.

3.63.6.9 Congratulations/Completion Pages

- When installing a single program interactively, start the program (and close the setup wizard) to indicate successful setup, instead of displaying a completion page. Exceptions: Setups that are run from the command line should not start programs.
  - Automatic updates (for example, Windows Update) shouldn't start programs.
  - Group policy installation shouldn't start programs.
  - Any IT professional setup scenarios (because they are not installing for their own use).
  - **If the setup has follow-up steps after installation, list them on a Completion page.** But to justify a Completion page, make sure users are likely to perform the steps, and that the steps genuinely need to be stated (that is, they are not obvious).
- When installing a suite of programs, display a Completion page to indicate success and any follow-up steps that may be necessary.
- Leaving users in control
- Don't clutter the Start menu with entries for uninstall, readme files, Help files, or links to Web sites. For more guidelines, see Start menu.

- Don’t gather personal information, such as that used for marketing purposes. Setup isn’t an opportunity to push your own agenda, cross-sell other program offerings, or conduct market research; you can damage the trust relationship with your users this way.
- Don’t force users to opt out of installing optional features. Allow them to opt in instead. For example, users should explicitly choose to install a Windows Desktop Gadget.
- Allow users to add or remove optional features using the setup program after initial setup. Users can perform this task using the Uninstall or change a program control panel item.
- For customer experience improvement initiatives, explain what data is transmitted, how it is used, and how long it is kept. Use a link to a privacy statement Help topic for this purpose.
- Avoid using sound, because many installation scenarios are unattended, and because sound can be unnecessarily distracting even during attended installations.

3.63.7 Security

- For Internet-based setup, provide any security updates automatically during initial setup. Users should not have to update as a separate step.
- Avoid recommending that users turn off firewalls as a prerequisite to installing your program.
  - If a firewall must be turned off, do the following: Limit the duration of this condition to as short a time as possible.
  - Explicitly point out when users can turn the firewall back on again.

3.63.8 Uninstall

- Uninstall should remove all traces of a program, including the following:
  - Program files, including the setup program.
  - Start menu entries.
  - Desktop icons and Quick Launch icons (if any).
  - Registry settings.
  - File associations.
- Uninstall should leave behind the following:
  - User created files, such as document files.
  - Shared dynamic-link libraries stored in the System folder.

3.63.9 Help And Support

- Design your setup program not to need Help by asking clear, self-explanatory questions. Reserve Help for advanced questions that really benefit from further explanation.
- Don’t use readme files. These files are now obsolete and users don’t read them anyway. Instead, provide online content if needed.
- Link to appropriate Help topics or troubleshooting content from setup error messages. Make sure the Help content provides a clear path to resolving the problem. For more information, see Error Messages.
- Create log files to capture information useful to technical support. Don’t clutter the setup UI with technical support-related details that are meaningless to most users. Use log files for this purpose instead.

3.63.10 Text

- **Be concise.** Setup wizards often overexplain features and options, using blocks of text that are difficult to scan quickly. **Exceptions:** Spell out all acronyms. Setup is often users’ first experience with your program, so don’t assume they understand jargon such as acronyms.
- Explain unfamiliar terminology and concepts, preferably in place but using Help topics if necessary.
- **Prefer a friendly, professional tone; avoid an overly-technical tone.**
- Don’t use *now* in command button labels because the immediacy of the command can be taken for granted. **Exception:** When necessary, use *now* to differentiate commands that start a task from commands that perform a task immediately.
- Only one command in a task flow should be labeled with *now*. So, for example, a **Download now** command should never be followed by another **Download now** command.
- Use *license terms*, not *license agreement*, *licensing agreement*, *end user license agreement*, or *EULA*.

3.64 First Experience (Page 711)

3.64.1 Guidelines

3.64.1.1 General

Limit first experiences to tasks and settings required to use a program or feature, and only include these when there is no better alternative. See the previous table for alternatives. Exception: Add personalization or program customization settings to the first experience if their customization is part of the core experience or crucial to the user's personal identification with the program.

- **Use the setup experience** for tasks and settings if they apply to all users or changing settings requires elevation.
- **Use the first use experience** for tasks and settings if they apply to individual users.

3.64.2 Presentation

- **Prefer optional tasks and settings to required tasks and settings.** Avoid forcing users to configure your program.
- **Take optional tasks and settings out of the main task flow whenever practical.** For example, many setup programs provide a custom installation path to remove infrequently changed settings from the main task flow.
  - **Don't overwhelm users with tasks and settings: Start simple.** Begin with simple, personalization settings and progress towards more complex, technical tasks and settings. For example, Windows setup starts with personal information and ends with network configuration.
  - **Use a contextual first experience** for tasks and settings if they apply only to features that aren't fundamental to the main program.
  - **Don't present everything all at once.** Consolidate to use a single UI instead of multiple UI surfaces, or display tasks at different times instead of all at once.
- **Express questions and options in terms of users' goals and tasks, not in terms of technology.** Provide options that users understand and clearly differentiate. Make sure to provide enough information for users to make informed decisions.
- **If the need for personal information isn't obvious, explain why your program needs the information and how it will be used.**
- **Present first experiences full screen only if users can't productively perform other tasks.** For example, Windows setup is presented full screen to discourage users from performing other tasks while Windows is being installed. Most first experiences shouldn't be full screen.

3.64.3 Settings

- **For all settings, select the safest (to prevent loss of data or system access), most secure and private value by default.** If safety and security aren't factors, select the most likely or convenient value. Choosing good defaults is an effective way to simplify the first experience.
- Require users to opt in for:
  - Settings with legal implications, such as end user licensing agreements (EULAs). Such settings can't have default selections.
  - Features that perform automatic system configuration changes, such as Windows automatic updates.
  - Features that reveal personally identifiable information (PII) or system information.
  - Changes to the user's desktop beyond adding entries to the Start menu, such as adding icons to the desktop or quick launch bar.
  - Optional software, such as product enhancements, subscriptions, and third-party products.
- **If a setting is strongly recommended, add "(recommended)" to the label.** For radio buttons and check boxes, be sure to add to the control label, not the supplemental notes.
- **If a setting is intended only for advanced users, add "(advanced)" to the label.** For radio buttons and check boxes, be sure to add to the control label, not the supplemental notes.

3.64.4 Tasks

- **Help users pass waiting time productively.** If the waiting time is typically between one and two minutes, consider providing helpful information while users are waiting, such as a presentation of what is new during setup.

  If the waiting time is typically longer than two minutes, make it easy for users to perform other tasks. Display the estimated wait time, recommend that users do something else in the meantime, and make task completion obvious by changing the screen significantly.
- **Reconsider presenting tutorials during the first experience.** Most likely users want to use your program right away and are interested in tutorials at a later point.
- **Don't use feature advertisement notifications in the first experience.** Instead of using a **feature advertisement notification**, design the feature to be easier to discover in contexts where it is needed, or don't do anything special and let users discover the feature on their own.

- **Don't use any notifications during the initial Windows experience.** To improve its first experience, Windows 7 suppresses all notifications displayed during the first few hours of usage. Design your program assuming users won't see any such notifications.

3.65 Printing (Page 718)

3.65.1 Guidelines

3.65.1.1 General

- **Don't print blank pages or pages with just headers and footers.** However, print blank pages if the headers or footers contain page numbers and those page numbers might referenced elsewhere.
- **Completely spool out any pending print jobs before shutting down a program.**

3.65.2 Formatting Pages

- Reformat text layout to fit within the target page size. Never truncate text.
- If users don't control the format of the document: Automatically handle large objects by scaling, rotating, or splitting across pages.
- Optimize the page breaks to eliminate blank and nearly blank pages.
- Convert light text on a dark background to dark text on a white background.
- Remove backgrounds and other design elements, especially if they are unsuitable for a black and white printer.
- If your program presents separate partial documents, provide a printer-friendly format option to consolidate them into a single document for printing.
- Remove interactive elements: Remove navigation controls and command buttons.
- Make sure that all data is visible without scrollbars.
- Replace links with their text equivalent.

3.65.3 Oversized Objects

Handling large objects, such as spreadsheets, graphics, and photos, is a problem unique to printing. Choose one of the following approaches:

- **Scale the object to fit on the page.** This approach works well if the object is only slightly too large to print, keeping the object on a single page is important, and the object is still legible when scaled down.

- **Rotate the page.** This approach works well when a few pages print better in landscape mode when in portrait mode (and vice versa).
- **Print the object on several pages.** The approach works well when the object can't be scaled, or shouldn't be scaled, and rotating the page doesn't help or isn't wanted. If the object has internal boundaries (such as the column and row dividers in a spreadsheet), break the pages on these boundaries instead of within the content. Also, repeat any elements required to understand the page, such as legends or column headers. When splitting an object on several pages, assign the page numbers in reading order (left to right, top to bottom).
- **Truncate the object** (printing only the part of the object still visible after truncation). This approach is the simplest solution to implement, but likely to be the least acceptable. Also note that truncating is never acceptable for text.

3.65.4 Headers And Footers

- **Provide headers and footers for advanced and intermediate document creation programs.** Consider providing headers and footers for other types of programs if they are used for multipage documents.
  - **Make headers and footers customizable.** Allow users to define the left, center, and right portions. For headers, put the document name on the left side by default.
  - For footers, put the document copyright or source on the left side, and the page number on the right side, by default.
  - **Use friendly file path and URLs.** Display spaces as spaces, not "%20."

3.65.5 Print Commands

- For menu bars and shortcut menus, use the Print command that displays the Print options common dialog. Use an ellipsis to indicate that additional information is required.
- For toolbars used with a menu bar, use an immediate Print command. Clicking the button prints a single copy of the document to the default printer. Such toolbar commands should be immediate. To indicate that the command is immediate, put the default printer in the tooltip. Users can access the full Print command from the menu bar.
  - **For toolbars used without a menu bar, use a Print split button.** Clicking the button prints a single copy of the document to the default printer. Clicking the arrow portion of the button drops down a menu with full Print, Print preview, and Page setup commands.
- For the ribbon command user interface, put the Print command in the application menu.

3.65.6 Print Options

- **Don't create a custom Print options dialog box.** If you must provide additional options, extend the Print options common dialog. Don't use a separate dialog for additional print options.
- **When extending the Print options common dialog, don't duplicate any features already provided.**
- **If users are likely to maintain settings from one print job to the next, make those settings the defaults.** For the first print job after program launch, use the standard default values, including the default printer. For subsequent print jobs in the program **instance**, preserve the last selected printer and paper size. Don't preserve the number of copies or page ranges, because these are far less likely to be reselected later.
- **Optimize the settings by removing options that currently don't apply.** Remove options that are inconsistent with the capabilities of the selected printer or characteristics of the current document. For example, a photo printing application could limit the combinations of paper size, paper type, and print quality that give the best results, so choosing a glossy paper option might remove envelopes from the paper formats. If for any reason users want to see all the options, you can provide this ability through a control such as a check box.
- **For advanced document creation programs, save the document-related print options within the document itself.** For these programs, the print options are an integral part of the document.
- **For other types of programs, save settings on a per-user basis.**
- **Consider selecting a non-default printer for specialized printing.** For example, a photo printing application could always select the printer last used by the program, regardless of the system default printer. Doing so assumes that the system default printer isn't likely to be a photo printer. Such programs should save the setting for the last selected printer.
- Don't lock up your program while detecting printer capabilities. Doing so presents a poor user experience. Instead, either:
  - Perform the printer capability detection in a separate thread.
  - Time out after 10 seconds.
  - Provide a dialog box to allow users to cancel.

3.65.7 Print Previews

- **Provide a print preview feature whenever appropriate.** All document creation programs benefit from print previews, but users don't expect them in simple document creation programs. For advanced document creation programs, consider having print preview support directly within the main program window.

- Provide print preview features that allow users to: Evaluate margins, page breaks, page orientation, headers, and footers.
- Browse through all the pages.
- Print directly from the print preview.
- Consider providing an interactive print preview so that users can adjust frequently changed settings like margins and line breaks directly within the preview.
- **Have print preview pages render within one second.** It's better to have a print preview that renders quickly and is accurate enough to allow users to evaluate the print results than to have a completely accurate preview that renders slowly.
- **For advanced document creation programs, consider extending the standard Print dialog box by incorporating preview functionality directly within it,** rather than creating a separate dialog for it.
- **Provide an obvious button for closing preview mode.**

3.65.8 Printing Errors

Note: Once the print job has been spooled to the printer, Windows is responsible for any subsequent errors. Your program only has to handle errors that happen before the print job is spooled.

Before spooling a print job, check for any potential printing problems the user can fix. Present a clear, concise confirmation before continuing to print. Whenever possible, offer to fix the problem automatically. Doing so can prevent a waste of time and money.

3.65.9 Text

For the option to print on both sides of the paper, label the option Print double-sided. Don't use the phrase "Manual Duplex."

3.66 Windows Environment (Page 734)

3.66.1 Desktop

3.66.1.1 Guidelines

- If your users are very likely to use your program frequently, provide an option during setup to put a program shortcut on the desktop. Most programs won't be used frequently enough to warrant offering this option.

- Present the option unselected by default. Requiring users to select the option is important because once undesired icons are on the desktop, many users are reluctant to remove them. This can lead to unnecessary desktop clutter.
- If users select the option, provide only a single program shortcut. If your product consists of multiple programs, provide a shortcut only to the main program.
- Put only program shortcuts on the desktop. Don’t put the actual program or other types of files.

3.67 Start Menu (Page 738)

3.67.1 Program Names

There are many factors in choosing a program name, most significantly your program’s image, recognition, and branding. The following Start menu guidelines affect your program’s discoverability, which is especially important for programs that are not well known or that are used infrequently. Programs that are well known or used frequently have much more latitude in naming.

- Choose program names that are easy to browse: Use self-explanatory names so that users can understand the primary purpose of your program by its name alone.
- Use program names that alphabetize well. Start names with an alphabetic character, not a space, number, or symbol.
- Avoid putting a version number in a program name unless that is how users normally refer to your program.
- Choose program names that are easy to search: Use either unique names that are easy to remember, or names that include words for which target users are likely to search.
- Prefer individual words over compound words.
- Avoid names that are easy to misspell or are misspellings.
- Avoid names with jargon and made-up words.
- Use **title-style capitalization**.

3.67.2 Start Menu Files

- Put only program shortcuts on the Start menu. Don’t put shortcuts to the following items on the Start menu:
  - **Program uninstallers.** Users access uninstallers through the Programs control panel item.
  - **Help files.** Users access Help topics directly from your program.

  - **Control panel items.** Users access control panel items from the Control Panel home page.
  - **Program options.** Users access program options from the Options command.
  - **Utility programs.** Users access utilities from commands in the Tools menu.
  - **Readme files.** Reconsider the need for such files, because most users never look at them. If you do need a Readme file, let users access it from your setup program.
  - **Web sites.** Users access Web sites through appropriate links in your program, or Help for technical support sites.
- **Use only a single shortcut per program on the Start menu.** Don't put shortcuts in different locations, such as in the top level and in the Accessories folder. Don't put shortcuts to access specific tasks within the program.
- **Label the program shortcut using the program's name.** Don't use other labels or include additional information in the label, such as trademark symbols. Don't include the company name unless users associate the company name with the product. Avoid putting the version number in a program shortcut unless that is how users normally refer to your program.
- **During setup, don't provide an option to put the program shortcut in the Start menu.** Do this automatically.
- **During setup, don't provide an option to pin the program shortcut at the top of the** Start menu. Let users choose to do this manually. Programs can no longer pin themselves in Windows Vista® and later.
- Use **title-style capitalization**.

3.67.3 Start Menu Folders

- **Locate program shortcuts in the top level of All Programs.** The improved scalability of the Start menu in Windows 7 and Windows Vista makes programs easier to find at the top level. **Exceptions:** Use **Control Panel** to access control panel items. They don't have shortcuts in All Programs. Also use Control Panel to access troubleshooting programs.
- Use the **Accessories** folder if target users think of your program as an accessory, and it isn't part of the core user experience. For example, Windows Media Player is a core user experience (and therefore in All Programs), whereas Sound Recorder is not a core user experience (so it is in Accessories).
- Use the **System Tools** folder only if your program is a system maintenance utility. System maintenance utilities may appear in both System Tools and Control Panel.
- Use the **Administrative Tools** folder for programs for IT professionals.
- Don't use the **Maintenance** folder. It is reserved in Windows Vista for Backup and Restore Center, Help and Support, Problem Reports and Solutions, and Windows Remote Assistance.

- **Create a product folder only if your product is a collection of individual programs** (three or more), and users think of your product in terms of that collection.
- **Use only a single-level product folder. Exception:** Use a secondary folder only if the product is a collection of several programs (six or more), and two or more of these programs are considered secondary utilities.
- **Choose descriptive, yet concise folder names:** Use three words or fewer.
  - Don't include trademark symbols. Avoid putting a version number in a folder unless that is how users normally refer to your product. Put the version number in the Start menu infotip instead.
  - Use **title-style capitalization**.

  Don't use the term "folder" in folder names.

3.67.4 Start Menu Infotips

- Use Start menu infotips to describe the item concisely and list the primary tasks that users can perform with the item.
- Be helpful. Focus on what users can do. Don't just repeat the item name or even use it in the description at all.
- Be specific. Avoid generic verbs and catch-all phrases like "and other tasks" and "and other kinds of documents". If the information is important, list it specifically; otherwise, assume that users understand that not everything is listed in the infotips.
- Be concise. Use 25 words or less. Longer infotips discourage reading.
- Start with a present-tense, imperative verb such as "create, edit, show," and "send". Prefer specific verbs over generic verbs such as "manage" and "open."
- Get right to the point. Don't use verbs that apply to any Start menu item, such as "start, lets you, use to," and "provides".
- Don't use language that sounds like marketing.
- Use sentence-style capitalization.

3.68 Taskbar (Page 744)

3.68.1 Guidelines

3.68.1.1 Taskbar Buttons

- Make the following window types appear on the taskbar (for Windows 7, by using a taskbar button thumbnail):

  - Primary windows (which includes dialog boxes without owners)
  - Property sheets
  - Modeless progress dialog boxes
  - Wizards
- For Windows 7, use taskbar button thumbnails to group the following window types with the primary window taskbar button it was launched from. Each program (specifically, each program perceived as a separate program) should have a single taskbar button.
  - Secondary windows
  - Workspace tabs
  - Project windows
  - MDI child windows
- **Restoring a primary window should also restore all its secondary windows,** even if those secondary windows have their own taskbar buttons. When restoring, place secondary windows on top of the primary window.
- **For Windows 7, programs that normally have desktop presence may temporarily display a taskbar button to show status.** Do so only if your program is normally displayed on the desktop and users frequently interact with it. A program that normally runs without desktop presence should use its notification area icon instead, even though it might not always be visible.

3.68.2 Icons

- **Design your program icon to look great on the taskbar.** Ensure it is meaningful, and reflects its function and your brand. Make it distinct, make it special, and ensure it renders well in all icon sizes. Spend the time necessary to get it right. Follow the **Aero-style icon guidelines**.
- **If your program uses overlay icons, design your program's base icon to handle overlays well.** Overlay icons are displayed in the lower right corner, so design the icon so that area can be obscured.
- **Don't use overlays in your program's base icon,** whether your program uses overlay icons or not. Using an overlay in the base icon will be confusing because users will have to figure out that it's not communicating status.

3.68.3 Overlay Icons

- **Use overlay icons to indicate useful and relevant status only.** Consider the display of an overlay icon to be a potential interruption of the user's work, so the status change must be important enough to merit a potential interruption.
- **Use overlay icons for temporary status.** The overlay icons lose their value if displayed constantly, so normal program status should not show an icon. Remove the overlay icon when the icon:

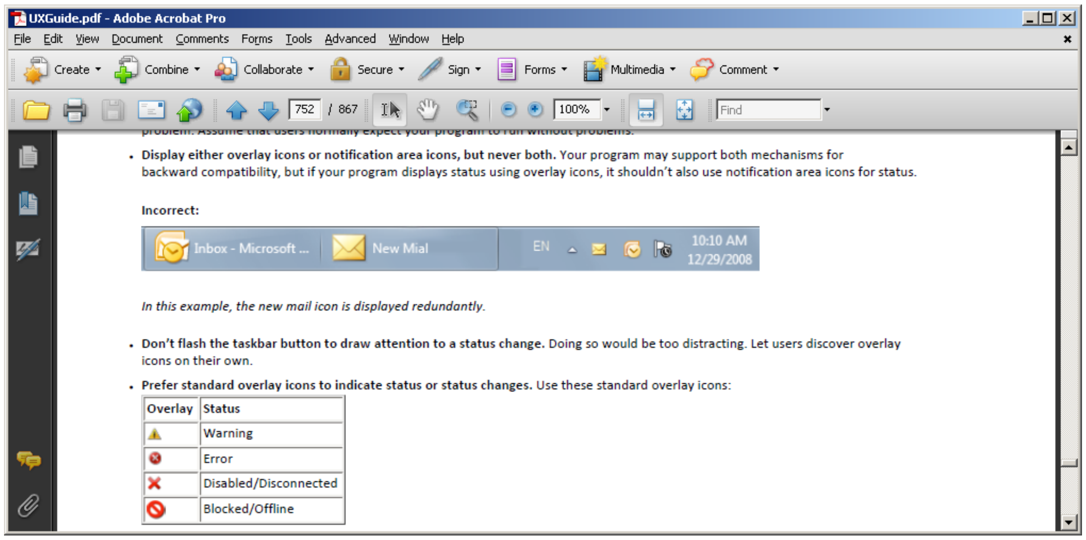
  - **Is for a problem:** Remove the icon once the problem has been resolved.
  - **Alerts that something is new:** Remove the icon once the user has activated the program.

- **Don’t display an icon to indicate that a problem has been solved.** Instead, simply remove any previous icon indicating a problem. Assume that users normally expect your program to run without problems.

- **Display either overlay icons or notification area icons, but never both.** Your program may support both mechanisms for backward compatibility, but if your program displays status using overlay icons, it shouldn’t also use notification area icons for status.

- **Don’t flash the taskbar button to draw attention to a status change.** Doing so would be too distracting. Let users discover overlay icons on their own.

- **Prefer standard overlay icons to indicate status or status changes.** Use these standard overlay icons:


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
752 / 867 100% Find
Inbox - Microsoft ...
New Mial
EN
10:10 AM
12/29/2008

- **Display either overlay icons or notification area icons, but never both.** Your program may support both mechanisms for backward compatibility, but if your program displays status using overlay icons, it shouldn’t also use notification area icons for status.

Incorrect:

In this example, the new mail icon is displayed redundantly.

- **Don’t flash the taskbar button to draw attention to a status change.** Doing so would be too distracting. Let users discover overlay icons on their own.
- **Prefer standard overlay icons to indicate status or status changes.** Use these standard overlay icons:

Overlay	Status
⚠	Warning
⊗	Error
✗	Disabled/Disconnected
⊘	Blocked/Offline

Figure 3-90: Standard overlay icons

- **For custom overlay icons, choose an easily recognizable design.** Use high-quality 16 × 16 pixel, full color icons. Prefer icons with distinctive outlines over square or rectangular shaped icons. Apply the other **Aero-style icon guidelines** as well.

- **Keep the design of custom overlay icons simple.** Don’t try to communicate complex, unfamiliar, or abstract ideas. If you can’t think of a suitable custom icon, use a standard icon error or warning icon instead when appropriate. These icons can be used effectively to communicate many types of status.

  - **Don’t change status too frequently.** Overlay icons shouldn’t appear noisy, unstable, or demand attention. The eye is sensitive to changes in the peripheral field of vision, so status changes need to be subtle.

- **Don’t change the icon rapidly.** If underlying status is changing rapidly, have the icon reflect high-level status.
- **Don’t use animations.** Doing so is too distracting.
- **Don’t flash the icon.** Doing so is too distracting. If an event requires immediate attention, use a dialog box instead. If the event otherwise needs attention, use a notification.

3.68.4 Taskbar Button Flashing

Use taskbar button flashing sparingly to demand the user’s immediate attention to keep an ongoing task running. It’s hard for users to concentrate while a taskbar button is flashing, so assume that they will interrupt what they are doing to make it stop. While flashing a taskbar button is better than stealing input focus, flashing taskbar buttons are still very intrusive. Make sure the interruption is justified, such as to indicate that the user needs to save data before closing a window. Inactive programs should rarely require immediate action. Don’t flash the taskbar button if the only thing the user has to do is activate the program, read a message, or see a change in status. **If immediate action isn’t required, consider these alternatives:**

- Use an **action success notification** to indicate that a task has completed.
- Do nothing. Just wait for users to attend to the issue the next time they activate the program. This is often the best choice.
- **If an inactive program requires immediate attention, flash its taskbar button to draw attention and leave it highlighted.** Don’t do anything else: don’t restore or activate the window and don’t play any sound effects. Instead, respect the user’s window state selection and let the user activate the window when ready.
- **For secondary windows that have a taskbar button, flash its button instead of the primary window’s taskbar button.** Doing so allows users to attend to the window directly.
- **For secondary windows that don’t have a taskbar button, flash the primary window’s taskbar button and bring the secondary window on top of all the other windows for that program.** Secondary windows that require attention must be topmost to ensure that users see them.
- **Flash only one taskbar button for one window at a time.** Flashing more than one button is unnecessary and too distracting.
- **Remove the taskbar button highlight once the program becomes active.**
- **When the program becomes active, make sure there is something obvious to do.** Typically, this objective is accomplished by displaying a dialog box that asks a question or initiates an action.

3.68.5 Quick Launch Shortcuts

Put program shortcuts in the Quick Launch area only if users opt in. Because Quick Launch was removed from Windows 7, programs designed for Windows 7 shouldn’t add program shortcuts to the Quick Launch area or provide options to do so.

3.69 Jump Lists (Page 753)

3.69.1 Design

Design Jump Lists to satisfy your users’ goals for their everyday tasks. Consider: **Your program’s purpose.** Think about what users are most likely to do next. For document creation programs, users are likely to return to recently used documents. For programs that show existing content, users may want access to resources they use frequently. For other programs, users might be likely to do tasks they haven’t done before, such as read new messages, watch new videos, or check their next meeting.

What users care about most. Think about why users would use the Jump List instead of other means. For example, users are more likely to care about destinations they explicitly identified as important (such as Web addresses users placed on their links bar or in Favorites, or typed in). They are less likely to care about those obtained indirectly or with little effort (such as Web addresses visited through redirection or by clicking links).

- **Don’t make destinations too granular.** Making destinations too narrow and specific can result in redundancy, with several ways to go to the same place. For example, instead of listing individual Web pages, list top-level home pages instead; instead of listing songs, list albums.
- **Don’t fill all the available Jump List slots if you don’t need to.** Focus Jump List content on the most useful items—if your program has only three useful items, provide only three. The more items in a Jump List, the more effort required to find any specific item.
- **Provide tooltips only when needed to help users understand Jump List items.** Avoid redundant tooltips because they are an unnecessary distraction.

3.69.2 Jump List Features vs. Program Features

- **Don’t make destinations and commands available only through Jump Lists.** The same destinations and commands should be available directly from the program itself.
- **Use consistent names for destinations and labels for commands.** Jump List items should be labeled the same as the equivalent items accessed directly from the program.

- **Enable your program to handle destinations and commands even when the program isn’t running.** Doing so is necessary for a consistent, dependable, and convenient experience.

3.69.3 Grouping

- **Provide at least one and at most three groups.** Jump List items are always grouped to label their purpose. Having more than three groups makes items harder to find.
- **Use the following standard group names when appropriate.** Standard group names are easier for users to understand.
- Recent Frequent Commands are given the Tasks group name, which is assigned by Windows and therefore can’t be changed.

3.69.4 Commands

Provide a fixed set of commands regardless of program running state, current document, or current user. The commands should apply to the entire program, not to a specific window or document. Doing so is necessary for a consistent, dependable, and convenient experience. Commands shouldn’t be removed or disabled. **Exceptions:** You may substitute or remove commands when:

- A set of mutually exclusive commands share a single command slot, as long as one command always applies.
- Commands don’t apply until specific features have been used, as long as the commands otherwise always apply.
  - **Use the following standard command labels when appropriate.** Standard command labels are easier for users to understand.
  - Sign in/Sign out New <object name> Play <object name> Go to <specific object name> Start Sync **Present the commands in a logical order.** Common orders include by frequency of use or order of use. Place highly related commands next to each other. Within the Tasks group, put separators between groups of related commands as needed.
  - **Don’t provide commands for opening or closing the program.** These commands are built into all Jump Lists.

3.69.5 Command Icons

Within the Tasks group, provide a command icon only when it helps users understand, recognize, or differentiate commands, especially when there is an established icon for the command used within the program. Exception: If your program uses both destinations (which always have icons) and commands, consider providing icons for all commands if not doing so would look awkward.

3.69.6 Destinations

- Provide a dynamic set of destinations that are specific to the current user, but independent of the program running state or current document. As mentioned previously, make sure they fit your program's purpose, are what users care about the most, and have the right level of specificity.
  - **When suitable, use an "automatic" destination list.** Automatic destinations are managed by Windows, but your program controls the specific destinations that are passed on. Consider using Recent for document creation programs where users are likely to return to recently used destinations.
- Consider using Frequent for programs that show existing content, where users are likely to return to items that they use often. Frequent destinations are sorted in order of frequency, most frequent first.
- Use Frequent if Recent would result in many useless destinations. Frequent lists are more stable, and the better choice when users go to many different destinations, but aren't likely to return to rarely used ones.
- If Recent or Frequent are equally suitable choices, use Recent because that approach is easier for users to understand and is more predictable.
  - If using Recent, and the program has an equivalent in the File menu, make the lists have the same contents in the same order. To users, these should appear to be the same lists.
  - **When necessary, use a custom destination list.** Your program has complete control over a custom destination list's contents and sort order, and therefore can base the list on any factors. Create custom versions of Recent or Frequent if those are suitable, but the automatic management doesn't work well for your program. For example, your program may need to track a variety of factors beyond open file commands. In this case, use the same name (Recent or Frequent) and sort order because users won't be aware of the difference.
  - Otherwise, use a different type of destination to better satisfy your user's goals. Often, these lists help users perform tasks that they haven't done before, such as read new messages, watch new videos, or check their next meeting.
  - Choose a sort order that corresponds to the user's mental model of the list. For example, a to-do style list would have the next thing to do listed first. If there is no clear mental model, sort the destination list in alphabetical order.
  - **Don't use multiple destination lists that give different views of the same data.** Rather, multiple destination lists should have mostly different data to support difference scenarios. For example, you can provide a Recent list or a Frequent list, but not both. Doing so is wasteful if overlapping items are present, but confusing if overlapping items are removed.
- If your program has a command to clear data for privacy, clear the Destinations lists as well. Destination lists may contain sensitive data.

3.69.7 Thumbnail Toolbars

3.69.7.1 Interaction

- Provide up to seven of the most important, frequently used commands that apply to the window shown in the thumbnail. Don't feel obligated to provide as many commands as you can—if your program has only three important, frequently used commands, provide only three.
- Use commands that are direct and immediate. These commands should have an immediate effect—clicking the command should not display a drop-down menu or dialog box for more input.
- Disable commands that don't apply to the current context, or that would directly result in an error. Don't hide such commands because doing so makes the toolbar presentation unstable.
- Don't dismiss the thumbnail when users click a command if they are likely to review the results or immediately click another command. Remove the thumbnail for commands that indicate that the user is finished for now, such as with commands that display other windows.

3.69.7.2 Presentation

- **Make sure thumbnail toolbar icons conform to the Aero-style icon guidelines.** For each command, provide high-quality 16 × 16, 20 × 20, and 24 × 24 pixel, full color icons. The larger versions are used in high-dpi display modes.
- **Make sure the icons are clearly visible against the toolbar background color in both normal and hover states.** Always evaluate icons in context and in the high-contrast modes.
- **Choose command icon designs that clearly communicate their effect.** Well-designed command icons are self-explanatory to help users find and understand commands efficiently.
- **Choose icons that are recognizable and distinguishable.** Make sure the icons have distinctive shapes and colors. Doing so helps users find the commands quickly, even if they don't remember the icon symbol. After initial use, users shouldn't have to rely on tooltips to distinguish between the commands.
- **Provide a tooltip to label each command.** A good tooltip labels the unlabeled control being pointed to. For guidelines and examples, see **Tooltips and Infotips**.

3.69.8 Progress Bars

- **Follow the general progress bar guidelines,** including not restarting or backing up progress, and using a red progress bar to indicate a problem.

- **Avoid using indeterminate progress bars.** Indeterminate progress bars show activity, not progress. Reserve indeterminate progress bars for those rare situations where users don't take activity for granted.

3.69.9 Deskbands

Note: Deskbands are no longer recommended for Windows 7. Use a taskbar button with a thumbnail toolbar instead. Your program may support both mechanisms for backward compatibility.

- **Display deskbands only if users opt in.** Offer to show deskbands in your program's setup and properties, but the option must be turned off by default.
- **Keep deskbands compact and simple.** Don't include any features directly on deskband windows that aren't accessed by most users in most scenarios. However, you can use menus to access less commonly used features.
- **Don't display programs in the taskbar that minimize to deskbands.** Minimize to a taskbar button or a deskband, but not both.
- **Make sure that deskbands use screen space efficiently in both horizontal and vertical orientations.** Doing so usually requires having orientation-specific layouts.

3.69.10 Text

3.69.10.1 Window Titles

When choosing window titles, consider the title's appearance on the taskbar:

- Optimize titles for display on the taskbar by concisely placing the distinguishing information first.
- For modeless progress dialog boxes, first summarize the progress. Example: "66% Complete."
- Avoid window titles that have awkward truncations.

3.69.11 Jump List Commands

- Start commands with a verb.
- Use sentence-style capitalization.

3.70 Notification Area (Page 766)

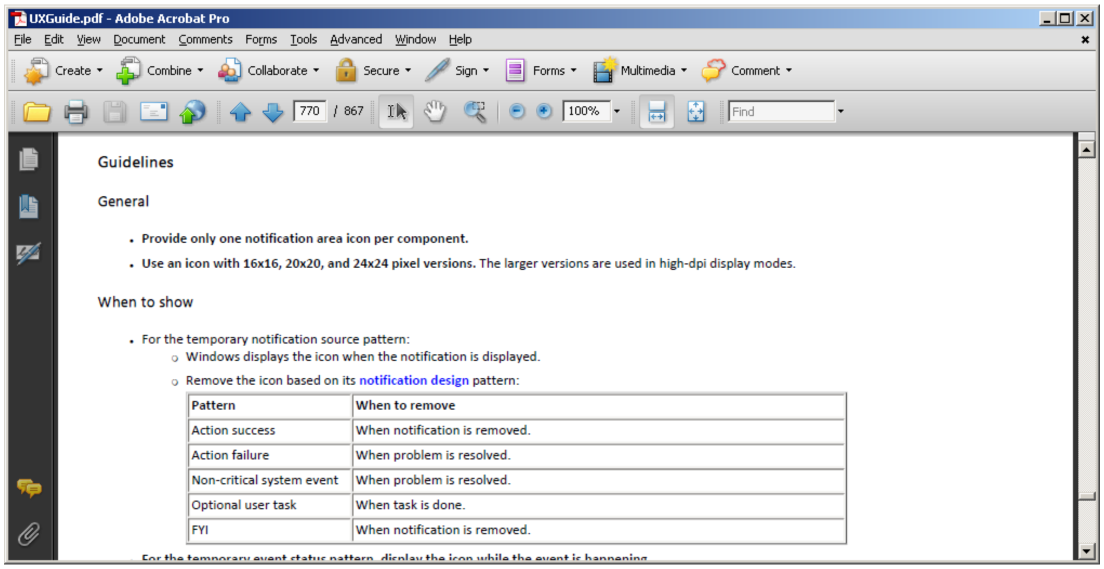
3.70.1 Guidelines

3.70.1.1 General

- Provide only one notification area icon per component.
- Use an icon with 16 × 16, 20 × 20, and 24 × 24 pixel versions. The larger versions are used in high-dpi display modes.

When to show

- For the temporary notification source pattern: Windows displays the icon when the notification is displayed.
- Remove the icon based on its notification design pattern:


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
770 / 867
100%
Find

Guidelines

General

- **Provide only one notification area icon per component.**
- **Use an icon with 16x16, 20x20, and 24x24 pixel versions.** The larger versions are used in high-dpi display modes.

When to show

- For the temporary notification source pattern:
  - Windows displays the icon when the notification is displayed.
  - Remove the icon based on its notification design pattern:

Pattern	When to remove
Action success	When notification is removed.
Action failure	When problem is resolved.
Non-critical system event	When problem is resolved.
Optional user task	When task is done.
FYI	When notification is removed.

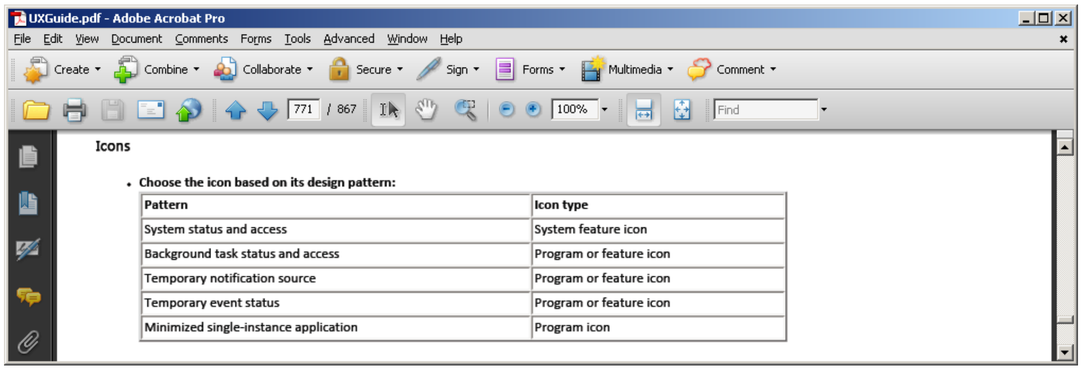
- For the temporary event status pattern, display the icon while the event is happening

Figure 3-91: Guidelines

- For the temporary event status pattern, display the icon while the event is happening.
- For all other patterns, display the icon when the program, feature, or process is running and the icon is relevant unless the user has cleared its Display icon in notification area option (for more information, see Context menus). Most icons are hidden by default in Windows 7, but can be promoted to the notification area by the user.
- Don’t display icons meant for administrators to standard users. Record the information in the Windows event log.

3.70.2 Where To Show

- Display windows launched from notification area icons near the notification area.


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
771 / 867
100%
Find

Icons

- Choose the icon based on its design pattern:

Pattern	Icon type
System status and access	System feature icon
Background task status and access	Program or feature icon
Temporary notification source	Program or feature icon
Temporary event status	Program or feature icon
Minimized single-instance application	Program icon

Figure 3-92: Icons

- **Choose an easily recognizable icon design.** Prefer icons with unique outlines over square or rectangular shaped icons. Keep the designs simple—prefer symbols over realistic images. Apply the other **Aero-style icon guidelines** as well.
- **Use icon variations or overlays to indicate status or status changes.** Use icon variations to show changes in quantities or strengths. For other types of status, use the following standard overlays. Use only a single overlay, and locate it bottom-right for consistency.
- **Avoid swaths of pure red, yellow, and green in your base icons.** To avoid confusion, reserve these colors to communicate status. If your **branding** uses these colors, consider using muted tones for your base notification area icons.
- For **progressive escalation**, **use icons with a progressively more emphatic appearance as the situation becomes more urgent.**
  - **Don't change status too frequently.** Notification area icons shouldn't appear noisy, unstable, or demand attention. The eye is sensitive to changes in the peripheral field of vision, so status changes need to be subtle.
  - **Don't change the icon rapidly.** If underlying status is changing rapidly, have the icon reflect high-level status.
  - **Don't use long-running animations to show continuous activities.** Such animations are a distraction. An icon's presence in the notification area sufficiently indicates continuous activity.
    - Brief, subtle animations are acceptable to show progress during important temporary, transitive status changes.
  - **Don't flash the icon.** Doing so is too distracting. If an event requires immediate attention, use a dialog box instead. If the event otherwise needs attention, use a notification.

- **Don’t disable notification area icons.** If the icon doesn’t currently apply, remove it. However, you can show an enabled icon with a disabled status overlay if users can enable from the icon.

Note: The following click events should occur on mouse up, not mouse down.

3.70.3 Hover

Display a tooltip or infotip that indicates what the icon represents.

3.70.4 Left Single-Click

Display whatever users most likely want to see, which may be:

- A flyout window, dialog box, or program window with the most useful settings and commonly performed tasks.
- A status flyout.
- The related control panel item.
- The context menu.

Users expect left single-clicks to display something, so not displaying anything makes a notification area icon appear unresponsive.

- **Display a context menu only if the other choices don’t apply**, with the default command in bold. In this case, display the same context menu that is shown on right-click to avoid confusion.
- **Prefer using a popup window over a dialog box** for a more lightweight feel. Show only the most common settings and have them take immediate effect for a simpler interaction. Dismiss the popup window if the user clicks anywhere outside the window.
- **Display small windows near the associated icon.** However, large windows such as control panel items can be displayed in the center of the default monitor.

3.70.5 Left Double-Click

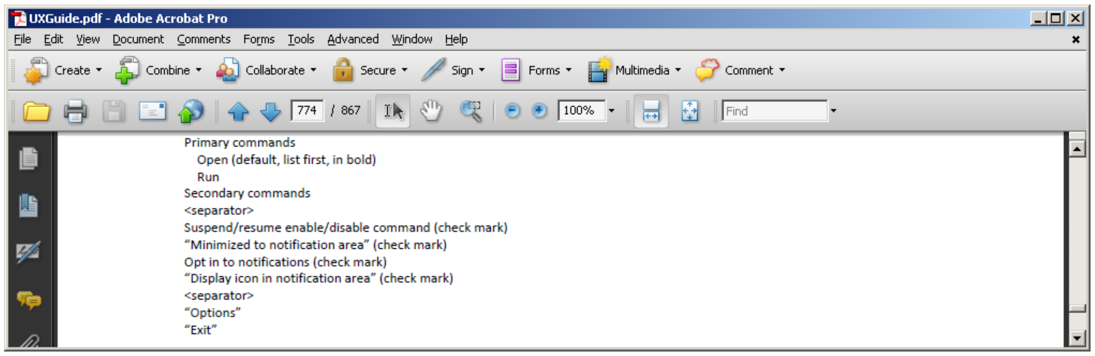
- **Perform the default command on the context menu.** Typically, this displays the primary UI associated with the icon, such as the associated control panel item, property sheet, or program window.
- **If there is no default command, perform the same action as a left single-click.**

3.70.6 Right-Click

Display the context menu, with the default command in bold.

3.70.7 Context Menus

- Display the context menu near its associated icon, but away from the taskbar.
- The context menu may include the following items, as appropriate, in the listed order (exact text is in quotes):


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
774 / 867
100%
Find

Primary commands
Open (default, list first, in bold)
Run
Secondary commands
<separator>
Suspend/resume enable/disable command (check mark)
"Minimized to notification area" (check mark)
Opt in to notifications (check mark)
"Display icon in notification area" (check mark)
<separator>
"Options"
"Exit"

Figure 3-93: Primary commands

- Remove rather than disable any context menu items that don't apply.
- For Open, Run, and Suspend/Resume commands, be specific about what is being opened, run, suspended, and resumed.
- Use Suspend/Resume running background tasks, Enable/Disable for everything else.
- Use check marks to indicate state. List and enable all states and place the check mark next to the current state. Don't disable options or change option labels to indicate the current state.
- All background tasks must have a Suspend/Resume command. Choosing the command should temporarily suspend the task. Users may want to temporarily suspend background tasks to increase system performance or reduce power consumption. Suspended background tasks are restarted when resumed by the user or when Windows is restarted.
- Allow users to opt into or out of different notification types if your program has notifications that some users might not want to see. The FYI notification pattern requires users to opt in, so these notifications must be disabled by default.
- Clearing the "Display icon in notification area" option removes the icon from the notification area, but doesn't affect the underlying program, feature, or process. Users can redisplay the icon from the program's Options dialog box. Don't automatically re-display the icon when Windows is restarted.
- The Exit command quits the program for the current Windows session and removes the icon. Don't have an Exit command if program can't be shut down. The program is restarted when Windows is restarted. Users can permanently quit the program from the Options dialog box.

  - **Don't have an About command.** Such information should be communicated by the icon, its infotip, and the context menu. If users want more information, they can view the primary UI. **Exception:** You may provide an About command if the icon is for a program that doesn't have desktop presence.

3.70.8 Rich Tooltips

- **Use rich tooltips only to make the information easier to understand.** Don't use rich tooltips just to decorate the feature. If you can't use richness to make the information easier to understand, use a plain tooltip instead.
- **Use a concise presentation.** Use concise text and a concise layout with a 32 × 32 pixel icon. Spacious tooltips risk being distracting, especially when displayed unintentionally.
- **Don't put controls or elements that appear interactive in a rich tooltip.** Tooltips aren't interactive and therefore shouldn't appear interactive. Don't use blue or underlined text.

3.70.9 Notification Area Flyouts

When appropriate, present notification area flyouts with three sections:

- **Summary.** Display the same information that is displayed in the icon's tooltip or infotip, possibly with more detail. For consistency, use the same text and icons, and generally the same layout (if using a rich tooltip). Unlike the infotips, this information is accessible when using touch.
- **Common tasks.** Present the most commonly performed tasks directly in the flyout.
- **Related links.** Provide at most one of each type of the following optional links:
  - A link to the most frequently performed task in Control Panel. Provide if there is a frequently performed task that can't be presented in the common tasks section.
  - A link to the related Control Panel item. This Control Panel item should allow users to perform any tasks that can't be performed in the common tasks section.
  - A link to a specific, relevant Help topic.

3.70.10 Options Dialog Box

- Options not accessible directly from the context menu must be in the Options dialog box. This dialog could be the feature's control panel.
- **The Options dialog box may include the following items** as appropriate (exact text is in quotes):

- Enable [feature name] (check box) Clearing this option permanently quits the program. Program can be restarted from its control panel item. The Exit command in the context menu quits the program only for the current Windows session.
- "Display icon in notification area" (check box) Removing the icon from the notification area doesn't affect the underlying feature.

  Selecting this option allows user to restore the icon, which of course can't be done from the icon itself.

- Disable features that are rarely used, or potentially annoying or distracting. Let users **opt in** to such features.

3.70.11 Minimizing Programs To The Notification Area

Note: Minimizing program windows to the notification area is no longer recommended for Windows 7. Use regular taskbar buttons instead. Your program may support both mechanisms for backward compatibility.

- To reduce taskbar clutter, consider providing the ability to minimize programs to the notification area only if all of the following apply:
  - The program can have only a single instance.
  - The program is run for an extended period of time.
  - The icon shows status.
  - The icon can be a notification source.
  - Doing so is optional and users must **opt in**.

Use the Minimize button on the application's title bar, not the Close button.

3.70.12 Text

3.70.12.1 Infotips

- The icon infotip should have one of the following formats (where the company name is optional):
  - (Company name) Feature, program, or device name
  - (Company name) Feature, program, or device name - Status summary
  - (Company name) Feature, program, or device name status statement.
  - (Company name) Feature, program, or device name Status list with each item on a separate line

3.70.12.2 Infotip Phrasing

- Focus on the most useful information. Display other information on left single-click.
- Be concise. Use sentence fragments or simple statements.
- Don't use ending punctuation unless tip is phrased as a complete sentence.
- Omit unnecessary words. Don't include the software version or other extraneous information.
- Don't explain how to interact with the icon.

3.71 Windows Desktop Gadgets (Page 784)

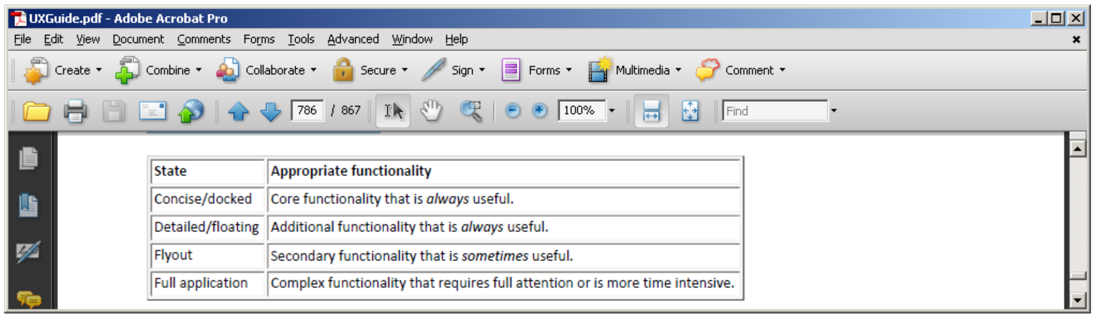
3.71.1 Guidelines

3.71.1.1 Controls

- **Have controls appear on hover.** In the default state, show only what's necessary. Most controls can be hidden until users hover the pointer over the gadget. This way you can draw attention to the most important parts of the gadget for both states.
- **Use tooltips to display all control labels** because there isn't enough space for labels.
- **Use controls that behave like common controls, but are themed appropriately for your gadget.** Choose colors that match the visuals of your gadget. Due to size constraints, you may need to make controls smaller than the standard common control sizes.
- **Use standard glyphs for commands.** If a command isn't represented by a standard glyph, create an appropriate glyph that has a consistent style. You may need to adjust the color and brightness of the glyphs depending on the gadget surface. All glyphs should have a rest, hover, and down state. They should commit on mouse up so that users can change their mind.
- **Avoid using scrollbars by default.** Instead, make default content fit comfortably within the gadget. For user customized content:
  - If necessary, use vertical scrollbars.
  - Consider using pagination if space is at a premium and it's not important for users to have complete control over content position.
  - Consider truncating content if doing so still provides useful information, such as the first few lines of an e-mail message. Show the complete content in a flyout or link to the associated application.
- Use Segoe UI, the Windows system font, for gadget text.

3.71.1.2 States

- **Expose core functionality in the concise/docked state.** Design gadgets assuming that they are always run in the concise/docked state. Consider the target audience, key scenarios, space constraints, and simplicity when determining core functionality.
- **Use the detailed/floating state to provide extra functionality, but don't do so gratuitously.** Link to your application or Web site for complex tasks that require full attention or are more time intensive, instead of trying to embed them in your gadget. If there is no need to add extra functionality, make the concise/docked and detailed/floating states the same.
- **Use the same drop shadow for both states to ensure that they align properly when placed along the screen edge.**
- **Use flyouts to display secondary information and functionality transiently.** If the information and functionality always needs to be available, put it directly on the gadget—don't put it in a flyout. Also, don't use flyouts for options; use an options dialog box instead. The following table compares the appropriate functionality for concise/docked gadgets, detailed/floating gadgets, flyouts, and full applications:


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
786 / 867
100%
Find

State	Appropriate functionality
Concise/docked	Core functionality that is *always* useful.
Detailed/floating	Additional functionality that is *always* useful.
Flyout	Secondary functionality that is *sometimes* useful.
Full application	Complex functionality that requires full attention or is more time intensive.

Figure 3-94: State and appropriate functionality

- **Use similar appearance and interaction for the concise/docked and detailed/floating states.** Users may be confused if the concise/docked and detailed/floating gadgets look and behave very differently. While the space constraints for the concise/docked state may require a more compact layout, try to keep the interactive components in roughly the same place. In both states, the same actions should have the same results.
- **On installation, consider having the gadget perform an action that demonstrates its purpose.** Doing so will make the gadget feel alive and self-explanatory. For example, a calendar gadget could turn the pages to today's date.
- **Avoid initial gadget configuration.** Choose the most likely or convenient default options. Requiring an initial configuration will make a gadget feel too heavy.
  - **Provide feedback for the loading and offline states.** Let users know that your gadget is offline or loading. For the loading state, use an **activity indicator** with text that explains what is loading, such as "Getting data...".

  - For the offline state, use a 16 × 16 pixel **information icon** with “Service not available” or other appropriate text. If necessary, make the gadget recognizable by showing faded placeholder content or cached data. Gadgets should activate automatically when a connection becomes available. While gadgets may reflect loss of connectivity, they don’t have to draw attention to it because users are likely to discover connectivity loss through traditional means, such as the network notification area icon or their Web browser.
- **Maintain state across sessions.** Users expect to find the gadget in the state they left it after last using their computer. For example, a partially solved puzzle should stay partially solved.

3.72 Interaction (Page 788)

- **Use the standard Windows pointer behavior.** For example, use the hand pointer only for links.
- **Don’t automatically change a gadget’s size.** Automatic size changes confuse and annoy users, because changing one gadget's size causes the other gadgets either to move (with the Windows Vista Sidebar), or potentially to overlap or go off-screen (with Windows 7). Never change size on hover. When necessary, you can change size on click.
- To the best extent possible, **design your gadget to eliminate the need for error handling and other types of messages.** These messages are contrary to the lightweight feel of gadgets.
- **If you must give an error, warning, or informational message, show the message in-place or as a different gadget state.** Don’t use dialog boxes for these messages. Show the message along with the appropriate 16 × 16 pixel **standard icon** for error and warning messages. Use no icon at all for minor user input problems.
- **Don’t provide Help for your gadget.** Make sure the design is self-explanatory instead.

3.72.1 Animation And Sound

- **Use animation judiciously.** Avoid gratuitous and distracting animation. The human eye is sensitive to motion, especially peripheral motion. When updating information, it is better to do it inconspicuously and wait for users to notice it at will. If you use animation to draw attention to something, make sure that attention is well deserved and worthy of interrupting the user’s train of thought. A transition, such as a cross fade, may make the update feel smoother, but it may also attract undue attention. Find the right balance between smoothness and impact by testing the gadget and tweaking variables such as animation speed and surface area. Animation that is triggered by a specific user action is encouraged. When used properly, it can make the gadget feel smooth and polished.

- **Use sound judiciously.** Don't distract users with sounds. Use sound sparingly and only on user interaction.

3.72.2 Options Dialog Boxes

- **Use an options dialog box only if needed.** Some gadgets don't need options. Useful options include personalization and feature customization, such as adding and removing content. Consider letting users adjust settings directly on the gadget if the options dialog box would duplicate much of the gadget. For example, users may expect to reorder items directly in a list by using drag-and-drop. Providing this functionality in the options dialog box might duplicate much of the gadget while making it harder for users to visualize the end result.
- **Don't provide options in flyouts.**
- **Provide access to the options dialog box through the Options button on the gadget.**
- **Make the options dialog box look like property windows**, but without tabs. Keep the options limited to a single page without a scroll bar. Follow the general **layout** guidelines for control placement in the dialog box.

3.72.3 Windows Integration

For Windows 7 programs that install gadgets:

- Provide an option during program setup to install the gadget.
- Present the option unselected by default.
- If users select the option, install only a single gadget.
- If the gadget is useful only after the program has been run by the user, install the gadget after first run instead of at setup. For example, a gadget that shows the most recent mail isn't relevant until users set up their mail account.
- Consider extending your gadget to **Windows SideShow**.

3.72.4 Recommended Sizing And Spacing

- **Gadgets in the concise/docked state are 130 pixels wide.** Include 5 pixels of drop shadow, 2 pixels on the left side and 3 on the right.
- **Gadgets in the concise/docked state should have a minimum height of 84 pixels and have a recommended maximum height of 200 pixels.** A gadget that uses space efficiently is more likely to be used.
- **Gadgets in the detailed/floating state should be no larger than 400 × 400 pixels.**

- **Options dialog boxes have a client area of 278 pixels wide and are no more than 400 pixels high.** The height should be adjusted to accommodate its controls without exceeding 400 pixels.
- **Options dialog box text is Segoe UI 12 pixels, with a line spacing of 14 pixels at 96 dpi.**

3.73 Control Panels (Page 792)

3.73.1 Guidelines

3.73.1.1 Property Sheet Control Panel Items

Don't use property sheets for new control panel items. Instead, use task flows to create a seamless experience and make full use of the categorization and search functionality of the control panel home page.

3.73.2 Task Flow Control Panel Items

3.73.2.1 General

Keep the most important content and controls visible without scrolling. Users won't scroll to see page content unless they have a reason to. You can make commit buttons always visible by placing them in a **command area** instead of the **content area**. Don't break up pages just to avoid scrolling. **You can vertically scroll long pages,** as long as the most important controls are visible without scrolling.

- **Don't use horizontal scrolling.** Instead, redesign the page content and use vertical scrolling. Pages may have horizontal scrollbars only when made very narrow.
- **To navigate between pages:** Use **task links** to start a task.
- Use task links or a Next button to navigate to the next page in a multi-step task.
- Use **commit buttons** to complete a task.
- Use the Back button in the menu bar to return to previously viewed pages. Do not add a Back button to the command area.
- Use the Address bar to return directly to the control panel home page.
- Use *See also* links in the task pane to navigate to pages in other control panel items. Otherwise, navigation should stay within a single control panel item.
- **Put only the control panel home page in the Address bar.** Clicking that link returns to the control panel home page, abandoning any work in progress without a **confirmation**.

- **Don't put a Close command button on control panel pages.** Users can close a control panel window using the Close button on the title bar.

3.73.2.2 Task Links And Buttons

- When a page has a small set of fixed options, use task links instead of a combination of radio buttons and a Next button. This allows users to select a response with a single click.
- You can put task links and buttons in the following places (in order of discoverability):
  - The **command area** (for command buttons on spoke pages only).
  - The **content area**:
    - Command buttons
    - Task links
    - Other links
- Links in the **task pane** (hub pages only).
- **Base the location of task links and buttons on importance and need for discoverability. Put only commit buttons in the command area.**
- **Put essential tasks in the content area.** Command buttons tend to draw the most attention, so reserve them for commands users must see. Task links also draw attention, but less than command buttons.
- **Reserve the task pane and plain links for secondary (less important) tasks.** The task pane is the least discoverable area of a task page, and plain links aren't as visible as command buttons and task links.
- For task links presented in the content area: **If there are more than seven links, group the links into categories.** Provide headings for each of the groups.
- **For fewer than seven links, present the links in a single group without a heading.**
- **Present task links and buttons in a logical order.** List task links vertically, command buttons horizontally.
- Within categories, **divide the commands into related groups.** Present the task groups by placing the most commonly used first, and within each group, place the most commonly used tasks first. **The resulting order should roughly follow the likelihood of use, but also have a logical flow. Exception:** Task links that result in doing everything should be placed first.
- **If there are many task links, give the most important tasks a more prominent appearance** by using a 24 × 24 pixel icon and two lines of text. For less important tasks, use a 16 × 16 pixel icon, or no icon, and a single line of link text.

- **Have clear physical separation between frequently used commands and destructive commands.** Otherwise, users might click destructive commands accidentally. You may need to reorder your commands somewhat to put destructive commands together.
- **Provide the mechanism to undo commands directly on the page.** Users shouldn't have to navigate somewhere else to undo a mistake.
- For task links, use either all default task link icons or all custom icons. Don't mix them. Consider using custom icons only if:
  - They aid users in comprehending the tasks.
  - They comply with the **Aero icon standards**.

  They have an unobtrusive appearance.

3.73.3 Dialog Boxes

When using task flows, you generally want a task to flow from page to page within a single window, but the following circumstances are exceptions.

- Use dialog boxes in the following circumstances:
  - To prompt users for an administrator user name and password. Always use the credential manager dialog box for this purpose. (Should be **modal**.)
  - To confirm an in-place command using a task dialog or message box. (Should be modal.)
  - To get input for in-place commands, such as for New, Add, Save As, Rename, and Print commands.
  - To perform secondary, stand-alone tasks. Using a **modeless**, secondary window allows such tasks to be performed independently and outside the main task flow.

3.73.4 Hub Pages

3.73.4.1 General

- Use task-based hub pages when:
  - There are a small number of commonly used or important tasks.
  - The configuration involves one or two objects (examples: monitors, keyboard, mouse, game controllers).
  - The configuration applies system-wide (examples: date and time, security, power options).
- Use object-based hub pages when:
  - **The configuration could involve several objects** (examples: user accounts, network connections, printers).

  - **The configuration applies only to the selected object**.

- Don’t use a hub page if the control panel item has a single page that contains all the tasks and properties involved.

3.73.4.2 Object Lists

- **List items in a logical order.** Sort named objects in alphabetical order, numbers in numeric order, and dates in chronological order.
- For object-based hubs, **provide object view commands in the task pane if the ability to change the view is important to the tasks**. The ability to change views is important if there are many objects and the default presentation doesn’t work well for all scenarios. Users can change the list view even if there aren’t explicit commands through the list view context menu, but it’s less discoverable.

3.73.4.3 Interaction

Don’t put commit buttons on hub pages. Hub pages are fundamentally launch points. Users never “commit” hub pages—they are never done with them. And commit buttons on hub pages make any tasks initiated from a hub confusing (users will wonder if those tasks need to be committed). **Exception:** If changing a setting requires **elevation**, provide an **Apply button** with a **security shield icon**. Disable the commit button once changes have been applied.

Consider putting the most useful properties directly on hub pages. Such hybrid hub pages are strongly recommended when users are most likely to use Control Panel to access those properties.

- Use an immediate commit model for any settings on hybrid hub pages so that changes are made immediately. Again, users never commit a hub page. If a setting requires a commit button, don’t put it on a hub page.
- Consider putting simple, “one-step” commands directly on hub pages instead of using navigation links.
- Confirm in-place commands whose effects cannot be easily undone. Use a task dialog or message box.
  - **For task-based hub pages, identify each task with a task link and an icon.** You can also provide an optional description for each link. However, try to make the task links self-explanatory and provide optional descriptions only to links that really need them.
  - For object-based hub pages, single-clicking selects objects, and double-clicking selects an object and navigates to its default page. The default page is typically a property page or a task-based hub page.
  - An object-based hub page may navigate to a task-based hub for the selected objects. However, such secondary hubs should be avoided because they make a control panel item feel too indirect.

3.73.5 Task Panes

Use task panes to present links to commands, views, and related control panel items.

- For task panes in task-based hubs, present links in the following order: **Secondary commands**. Present primary tasks only in the content area. Use the task pane for secondary, optional tasks. Consider a task primary if users must discover it in important scenarios; secondary if it's acceptable for users not to discover it.
  - **See also**. The optional links that navigate to related control panel items.
- For task panes in object-based hubs, present links in the following order:
  - **Object views**. The optional links used to control the presentation of the objects.
  - **Fixed commands**. The commands that are independent of the currently selected objects.
  - **Contextual commands**. The commands that depend on the currently selected objects, and are therefore not always displayed.
  - **See also**. The optional links that navigate to related control panel items.
  - **Don't use task panes in spoke pages.** Unlike hub pages, spoke pages should be focused on completing the task. You don't want to encourage users to navigate away before completion.

3.73.6 See Also Links

- Provide See also links in the task pane to help users find related control panel items, or the right control panel item if they have the wrong one. Link to items users are likely to associate with your control panel item.
- Link to a specific task page if that is what users are more likely to recognize. Otherwise, link to the entire control panel item. Use the control panel name without adding the phrase, "control panel".

3.74 Spoke Pages (Page 799)

3.74.1 General

- Use task pages for commonly used or important tasks where users need more guidance and explanation.
- Use form pages for features that have many settings and benefit from a direct, single-page presentation. The ideal tasks for such pages typically involve obvious changes to a few simple properties.
- Don't use task panes in spoke pages.

3.74.2 Interaction

- **Try to limit main tasks to a single page.** If more than one page is required, you can: **Use intermediate spoke pages for additional or optional steps.** Intermediate spoke pages are committed by the final spoke page.
- **Use independent windows for independent auxiliary tasks.** Independent windows are committed on their own, and independently of the main task.

  Doing so keeps the meaning of the commit buttons for the main task clear and unambiguous. Users should always be confident in understanding what they are committing to.

- **Don't use See Also links within a task flow.** These link to related, but different, control panel items. Although navigating to a different item is acceptable in hub pages, it is not in spoke pages, because doing so interrupts the task.
- **Don't use spoke pages for simple input or confirmations.** Use modal dialog boxes instead.

3.74.2.1 Interaction (Intermediate Spoke Pages)

- Use task links or a Next button to navigate to the next page. The way to proceed to the next step should always be obvious.
- You can have navigation links to optional task steps. To avoid confusion when users commit to the task, those extra pages should be intermediate pages within the same control panel item. They shouldn't have commit buttons, but should be committed when the main task is committed.

3.74.2.2 Interaction (Final Spoke Pages)

- **Use commit buttons to complete a task.** Use a **delayed commit model** for spoke pages, so that changes aren't made until explicitly committed (if users navigate away using Back, Close, or the Address bar, changes are abandoned). The commit buttons are a visual clue that the user is about to complete a task. Don't use links for this purpose.
- **Don't confirm commit buttons (including Cancel).** Doing so can be annoying. Exceptions:
  - The action has significant consequences and, if incorrect, not readily fixable.
  - The action may result in a significant loss of the user's time or effort.
  - The action is clearly inconsistent with other actions.
- **Don't confirm if users abandon changes** by navigating away using Back, Close, or the Address bar. However, you may confirm if a potentially unintended navigation may result in a significant loss of the user's time or effort.

- **Don't use command or navigation links** (including See also links). On final spoke pages, users should explicitly complete or cancel the task. Users should not be encouraged to navigate somewhere else, because doing so would likely cancel the task implicitly.
- **When users complete or cancel a task, they should be sent back to the hub page from which the task was launched.** If there is no such page, close the control panel window instead. Don't assume that spoke pages are always launched from another page.
- **Remove the stale "committed" pages from the Windows Explorer Back stack** when you return users back to the page that the task was launched from. Users should never see the pages that they have already committed to when clicking the Back button. Users should always make additional changes by completely redoing the task instead of clicking Back to modify stale pages.
- **Developers:** You can remove these stale pages using the ITravelLog::FindTravelEntry() and ITravelLogEx::DeleteEntry() APIs.

3.74.3 Commit Buttons

Note: Cancel buttons are considered to be commit buttons.

- **Confirm tasks using commit buttons that are specific responses to the main instruction, instead of generic labels such as OK.** The labels on commit buttons should make sense on their own. Avoid using OK because it isn't a specific response to the main instruction, and therefore easier to misunderstand. Furthermore, OK is typically used with modal dialog boxes and incorrectly implies closing the control panel item window.
- **Exceptions:**
  - Use OK for pages that don't have settings.
  - Use OK when the specific response is still generic, such as Save, Select, or Choose, as when changing a specific setting or a collection of settings.
  - Use OK if the page has radio buttons that are responses to the main instruction. To maintain the delayed commit model, you can't use task links on a final spoke page.
- **Provide a Cancel button to let users explicitly abandon changes.** While users can implicitly abandon a task by not confirming their changes, providing a Cancel button allows them to do so with greater confidence. **Exception:** Don't provide a Cancel button for tasks where users can't make changes. The OK button has the same effect as Cancel in this case.

- **Don’t use Close, Done, or Finish commit buttons.** These buttons are typically used with modal dialog boxes and incorrectly imply closing the control panel item window. Users can close the window using the Close button on the title bar. Also, Done and Finish are misleading because users are returned to the page where the task was launched from, so they aren’t really done.
- **Don’t disable commit buttons.** Otherwise users have to deduce why the commit buttons are disabled. It’s better to leave commit buttons enabled, and give a helpful error message whenever there is a problem.
- **Make sure the commit buttons appear on the page without scrolling.** If the page is long, you can make commit buttons always visible by placing them in a **command area**, instead of in the **content area**.
  - Right-align the commit buttons and use this order (from left to right): positive commit buttons, Cancel, and Apply.

3.74.4 Preview Buttons

- Make sure the Preview button means to apply the pending changes now but restore the current settings if users navigate away from the page without committing to the changes.
- You can use Preview buttons on any spoke page. Hub pages don’t need Preview buttons because they use an immediate commit model.
- Consider using a Preview button instead of an Apply button for control panel pages. Preview buttons are easier for users to understand and can be used on any spoke page.
- Provide a Preview button only if the page has settings (at least one) with effects that users can see. Users should be able to preview a change, evaluate the change, and make further changes based on that evaluation.
- Always enable the Preview button.

3.74.4.1 Live Previews

A control panel item has live preview when the effect of changes on a spoke page can be seen immediately.

- Consider using live preview for display settings when:
  - The effect is obvious, typically because it applies to the entire monitor.
  - The effect can be applied without significant delay.
  - The effect is safe and can be undone easily.
- **Use Save changes and Cancel for the commit buttons.** “Save changes” keeps the current settings, whereas Cancel reverts to the original settings. “Save changes” is used instead of OK to make it clear that any previewed changes haven’t been applied yet.

- **Don't provide an Apply button.** The live preview makes Apply unnecessary.
- **Restore any changes if users navigate away** using Back, Close, or the Address bar. To preserve changes, users must commit them explicitly.

3.74.5 Apply Buttons

- Make sure the Apply button means apply the pending changes (made since the task was started or the last Apply), but remain on the current page. Doing so allows users to evaluate the changes before moving on to other tasks.
- Use Apply buttons only on final spoke pages. Apply buttons should not be used on intermediate spoke pages to maintain an immediate commit model. Exception: You can use Apply buttons on a hybrid hub page if changing a setting requires elevation. For more details, see hub page interaction.
- Provide an Apply button only if the page has settings (at least one) with effects that users can evaluate in a meaningful way. Typically, Apply buttons are used when settings make visible changes. Users should be able to apply a change, evaluate the change, and make further changes based on that evaluation.
- Enable the Apply button only when there are pending changes; otherwise, disable it.
- Assign "A" as the access key.

3.74.6 Control Panel Integration

To integrate your control panel item with Windows, you can:

- Register your control panel item (including its name, description, and icon), so that Windows is aware of it.
  - If your control panel item is top level (see below): Associate it with the appropriate **category page**.
  - **Provide task links (including their name, description, keywords, and command line)** to indicate primary tasks and allow users to navigate directly to the tasks.
  - **Provide search terms** to help users find your task links using the Control Panel search feature.

3.74.7 Category Pages

- Add your control panel item to a category page only if:
  - Most users need it. Example: Network and Sharing Center
  - It is used many times. Example: System
  - It provides important functionality that isn't exposed elsewhere. Example: Printers

  - Control panel items that meet these criteria are referred to as *top level*.
- Don't add your control panel item to a category page if:
  - It is rarely used or used for one-time setup. Example: Welcome Center
  - It is targeted at advanced users or IT professionals. Example: Color Management
  - It doesn't apply to the current hardware or software configuration. Example: Windows SideShow (if not supported by current hardware).
  - Removing such control panel items from the category pages makes the top-level items easier to find. Given their usage, these control panel items are sufficiently discoverable through search or contextual entry points.
- Associate your top-level control panel item with the category under which users are most likely to look for it. This decision should be based on user testing.
- Consider associating your top-level control panel item with the second most likely category as well. You should associate a control panel item with two categories if users are likely to look for its main tasks in more than one place.
- Don't associate your control panel item with more than two categories. The value of the categorization is undermined if control panel items appear in several categories.

3.74.8 Task Links

- **Associate your control panel item with its primary tasks.** You can display up to five tasks on a Category page, but additional tasks are used for control panel searching. Use the same phrasing as you do for task links, possibly removing some words to make the task links more succinct.
- **Prefer to have task links lead to different places in your control panel item.** Having multiple links to the same place can be confusing.

3.74.9 Search Terms

- Register search terms for your control panel item that users are most likely to use to describe it. These search terms should include:
  - The features or objects configured.
  - The primary tasks.
  - These search terms should be based on user testing.
- **Also include common synonyms for these search terms.** For example, "monitor" and "display" are synonyms, so both words should be included.
- **Include alternative spellings or word breaks.** For example, users might search for either "web site" and "website." Consider providing common misspellings as well.

- **Consider singular vs. plural noun forms.** The control panel search feature doesn't automatically search for both forms; supply the forms for which users are likely to search.
- **Use simple present tense verbs.** If you register "connect" as a search term, the search feature won't automatically look for "connects," "connecting," and "connected."
- **Don't worry about case.** The search feature is not case-sensitive.

3.74.10 Standard Users And Protected Administrators

Many settings require administrator privileges to change. If a process requires administrator privileges, Windows Vista and later requires Standard users and Protected administrators to elevate their privileges explicitly. Doing so helps prevent malicious code from running with administrator privileges.

3.74.11 Schemes And Themes

A scheme is a named collection of visual settings. A theme is a named collection of settings across the system. Examples of schemes and themes include Display, Mouse, Phone and Modem, Power Options, and Sound and Audio Options.

- Allow users to create schemes when: Users are likely to change the settings.
- Users are most likely to change settings as a collection.
- Schemes are useful when users are in a different environment, such as a different physical location (country/region, time zone); using their computer in a different situation (on batteries, docked/undocked); or using their computer for a different function (presentations, video playback).
- Provide at least one default scheme. The default scheme should be well named and apply to most users in most circumstances. Users shouldn't have to create a scheme of their own.
- Provide a preview or other mechanism so that users can see the settings within the scheme.
- Provide Save As and Delete commands. A rename command isn't necessary—users can rename schemes by saving under the desired name and deleting the original scheme.
- If the settings can't be applied without a scheme, don't allow users to delete all the schemes. Users shouldn't have to create a scheme of their own.
- If the schemes are not completely independent (for example, power schemes depend upon the current laptop mode of operation), make sure there is an easy way to change settings that apply across all schemes. For example with power schemes, make sure that users can set what happens when a portable computer's lid is closed in a single location.

3.74.12 Miscellaneous

Use Control Panel extensions for features that replace or extend existing Windows functionality. The following control panel items are extensible: Bluetooth Devices, Display, Internet, Keyboard, Mouse, Network, Power, System, Wireless (infrared).

3.74.13 Default Values

- **The settings within a control panel item must reflect the current state of the feature.** Doing otherwise would be misleading and possibly lead to undesired results. For example, if settings reflect the recommendations but not the current state, users might click Cancel instead of making changes, thinking that no changes are needed.
- **Choose the safest (to prevent loss of data or system access) and most secure initial state.** Assume that most users won't change the settings.
- **If safety and security aren't factors, choose the initial state that is most likely or convenient.**

3.74.14 Text

3.74.14.1 Item Names

- **Choose a descriptive name that clearly communicates and differentiates what the control panel item does.** Most names describe the Windows feature or object being configured, and are displayed in the Classic View of the control panel home page.
- **Don't include the words "Settings," "Options," "Properties," or "Configuration" in the name.** This is implied, and leaving it off makes it easier for users to scan.
- **If the control panel item configures related features, list only those features required to identify the item, and list the features the most likely to be recognized or used first.**
- Use **title-style capitalization**.

3.74.14.2 Page Titles

Note: As with all Explorer windows, control panel page titles are displayed on the **address bar**, but not the title bar.

- For hub pages, use the control panel item name.
- For spoke pages, use a concise summary of the page's purpose. Use the page's main instruction if it is concise; otherwise use a concise restatement of the main instruction.

3.75 Task Link Text (Page 807)

The following guidelines apply to links to task pages, such as Category page task links and See also links.

- **Choose concise task names that clearly communicate and differentiate the task.** Users shouldn't have to figure out what the task really means or how it differs from other tasks. **Incorrect:** Adjust display settings **Correct:** Screen resolution. *In the correct example, the wording conveys more precision.*
- **Retain similar language between task links and the pages they point to.** Users shouldn't be surprised by the page that is displayed by a link.
- **For task pages, design the main instruction, commit buttons, and task links as a related set of text.**
- While tasks often start with verbs, **consider omitting the verb on Category pages if it is a generic, configuration-related verb that doesn't help communicate the task.**
- **If the task configures several related features, list only the features that are representative of the set.** Omit details that can be readily inferred.
- **You should phrase tasks in terms of technology only if target users would do so as well.**
- Use plural nouns only if the system can support more than one.
- Use **sentence-style capitalization**.
- Don't use ending punctuation.

3.75.1 Main Instructions

- **For the hub page, use the main instruction to explain the user's objective with the control panel item.** The main instruction should help users determine if they have selected the right control panel item. Keep in mind that users might have selected your control panel item in error and are actually looking for settings that are part of another control panel item.
- **For spoke pages, use the main instruction to explain what to do on the page.** The instruction should be a specific statement, imperative direction, or question. Good instructions communicate the user's objective with the page rather than focusing purely on the mechanics of manipulating it.
- **Use specific verbs whenever possible.** Specific verbs are more meaningful to users than generic ones.
- Use **sentence-style capitalization**.
- **Don't include final periods if the instruction is a statement.** If the instruction is a question, include a final question mark.

3.75.2 Supplemental Instructions

- For the hub page, use the optional supplemental instruction to further explain the purpose of the control panel item.
- For spoke pages, use the optional supplemental instruction to present additional information helpful to understanding or using the page. You can provide more detailed information and define terminology.
- Use complete sentences and sentence-style capitalization.

3.76 Page Text (Page 809)

- Don’t restate the main instruction in the content area.
- Use the word “page” to refer to the page itself.
- Use the second person (you, your) to tell users what to do in the main instruction and content area. Often the second person is implied.
- Use the first person (I, me, my) to let users tell the control panel item what to do in the content area that responds to the main instruction.

3.76.1 Task Links

- **Choose concise link text that clearly communicates and differentiates what the task link does.** It should be self-explanatory and correspond to the main instruction. Users shouldn’t have to figure out what the link really means or how it differs from other links.
- **Always start task links with a verb.**
- Use **sentence-style capitalization**.
- Don’t use ending punctuation.
- **If the task link requires further explanation, provide the explanation in a separate text control** using complete sentences and ending punctuation. However, add such explanations only when needed—don’t add explanations to all task links because another task link needs one.

3.76.2 Commit Buttons

- **Use specific commit button labels that make sense on their own and match the main instruction.** 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.
- **Always start commit button labels with a verb.**
- **Don’t use Close, Done, or Finish.** These button labels are better suited for other types of windows.

- Use **sentence-style capitalization**.
- Don't use ending punctuation.
- Assign a unique **access key**. **Exception:** Don't assign access keys to Cancel buttons, because Esc is its access key. Doing so makes the other access keys easier to assign.

3.77 Help (Page 816)

3.77.1 Guidelines

3.77.1.1 Entry Points

- **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.
- **Use Help links for dialog boxes, error messages, wizards, and property sheets.** If the Help link applies to specific controls, place it under them, left-aligned. If the Help link applies to the entire window, place it at the bottom left corner of the window's content area.
- **Use Help links instead of generic textual references to Help whenever technically possible.**
- **Use a Help button with the Help icon for the hub pages of control panel items.** Place it in the upper-right corner. These buttons don't have a label, but have a tooltip that reads *Help*.
- **F1 Help is optional.** Users have grown accustomed to finding Help information related to the immediate context of the UI on the screen by pressing the F1 key, which is labeled *Help* on standard keyboards. You can include F1 Help if, for example, usability studies show that your users expect to find it, or your program UI is complex enough to benefit from contextual assistance.
- **Programs with menu bars can have a Help menu category.**
- **For keyboard accessibility, provide tab stops for Help buttons and links.**
- Help button and link behavior should be as follows: Help pane opens and a dedicated Help topic is displayed; the UI that invoked the Help pane should remain open to preserve the contextual experience.

- **Don’t use the following obsolete Help entry point styles: “Learn more” or “Learn more about...” links, generic Help buttons, and context-sensitive Help buttons on the title bar.** Although they have been used in the past, usability studies have determined that users tend to ignore them. Use links to specific Help topics instead.
- Don’t use “Learn more” or “Learn more about...” links.
- Don’t use generic Help buttons.

3.77.2 Content

- **Don’t create obvious content.** Help topics that repeat what is in the primary UI don’t add value.
- **Don’t create content that the user can’t act on in some way. Exception:** Some conceptual content delivers important background information without necessarily leading to user action.
- **Avoid vague resolutions to problems.** For example, “contact your system administrator” or “reinstall the application” tend to frustrate users. **Exception:** Recommend contacting the system administrator if that is the only practical solution, and system administrators expect to be contacted for the problem.
- **Avoid content that addresses highly unlikely user scenarios.** Develop your main Help content for what you anticipate will be normal usage; note important exceptions to expected usage, but treat this content as a lower priority.
- **Gather feedback from your users about how helpful your Help topics are.** Allow users to rate individual topics. Conduct **usability studies** on your documentation to ascertain problems involving quality and discoverability of content. **Tip:** User feedback is also a great way to generate more task-based content, focused on what users are actually doing with your program, as opposed to feature-based content, focused simply on a description of the technology.
- **Provide multiple ways of accessing your content.** A table of contents, an index, and a **search** mechanism are three of the most common methods of improving discoverability.
- **If there is more than one way to perform a task, in most cases you can just document the most common way used by inexperienced users.**

3.77.3 Icons

Use the Help icon only for Explorer windows and the hub pages of control panel items. Don’t use the Help icon with Help links.

3.78 Text (Page 824)

3.78.1 Help Links

Provide specific information about the content of the Help topic, using as much relevant, concise text as necessary. Users often ignore generic Help links. Make sure the results of the link are predictable—users shouldn't be surprised by the results. Exception: You can use "More information" to supplement instructions that are directly in the UI, especially if providing specific information in the Help link leads to unnecessary repetition or makes the link less compelling.

- Whenever possible, phrase Help link text in terms of the primary question answered by the Help content. Don't use "Learn more about," "Tell me more about," or "Get help with this" phrasing.
- If the most relevant information can be summarized succinctly, put the summary directly in the UI instead of using a Help link. However, you can use a Help link to provide supplemental information.
- Phrase Help links to clearly indicate assistance. Help links should never read like action links.
- Use the entire Help link for the link text, not just the keywords.
  - **Exception:** Help links to external Web sites should simply use the name of the site or page as the link. Any text introducing the name of the site need not be included in the link itself.
  - Help links don't have to match Help topic headings exactly, but there should be a strong and obvious connection between the two. Design links and headings in pairs for this reason.
- If the Help content is online, make that clear in the link text. Doing so helps make the result of the links predictable.
  - Use complete sentences.
  - Don't use ending punctuation, except for question marks.
  - Don't use ellipses for Help links or commands.

3.78.2 Help Content

- Format UI elements using bold to make them easy to identify. This is especially useful for procedural Help topics, allowing users to scan through procedures and quickly see pertinent UI elements.
- Format captions using italic. This applies to tables, art, screenshots, and other graphic elements that benefit from brief textual explanation.
- Refer to Help simply as *Help*. Generally, don't use the phrase "online Help" unless you are in fact referring to content on your Web site.

3.79 User Account Control (Page 827)

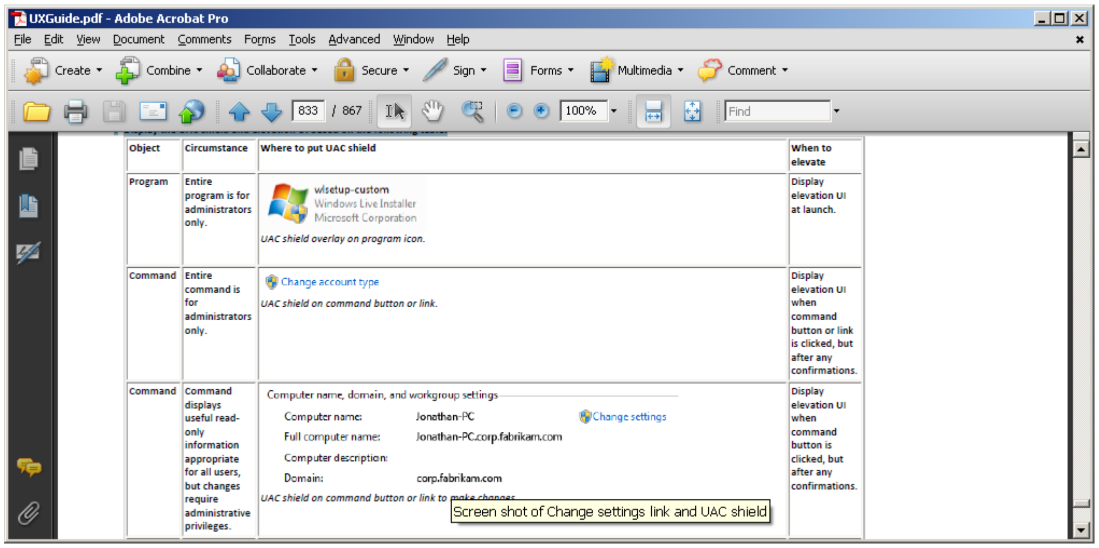
3.79.1 UAC Shield Icon

- Display controls with the UAC shield to indicate that the task requires *immediate* elevation when UAC is fully enabled, even if UAC isn't currently fully enabled. If all paths of a wizard and page flow require elevation, display the UAC shield at the task's entry point. Proper use of the UAC shield helps users predict when elevation is required.
- If your program supports multiple versions of Windows, display the UAC shield if at least one version requires elevation. Because Windows XP never requires elevation, consider removing the UAC shields for Windows XP if you can do so consistently and without harming performance.
- Don't display the UAC shield for tasks that don't require elevation in most contexts. Because this approach will sometimes be misleading, the preferred approach is to use a properly shielded contextual command instead.
- Because tasks don't remember elevated states, don't change the UAC shield to reflect state.
- Display the UAC shield even if User Account Control has been turned off or the user is using the Built-in Administrator account. Consistently displaying the UAC shield is easier to program, and provides users with information about the nature of the task.

3.79.2 Elevation

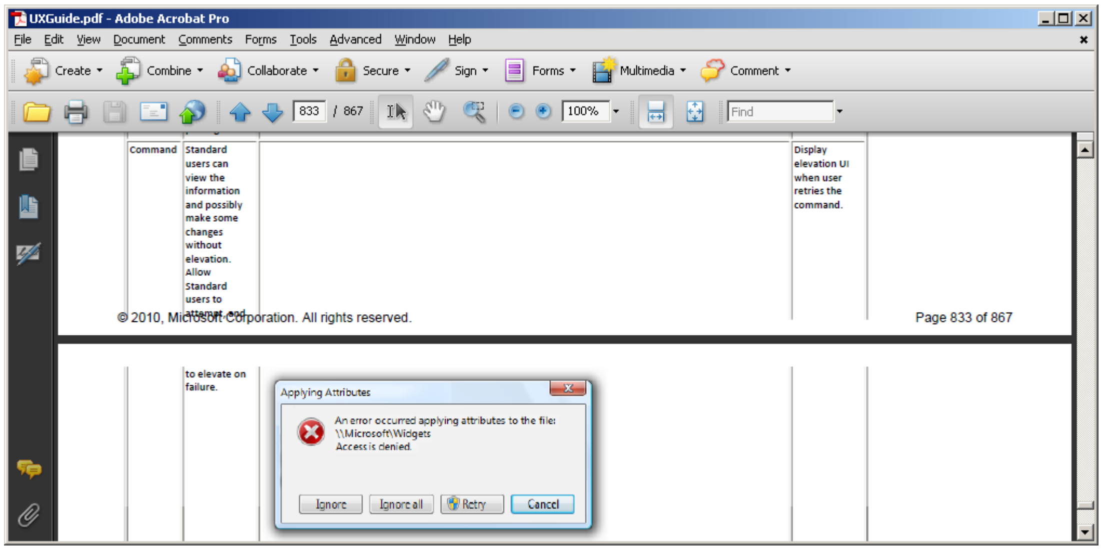
- Whenever possible, design tasks to be performed by Standard users without elevation. Give all users access to useful read-only information.
- Elevate on a per task basis, not on a per setting basis. Don't mix Standard user settings with administrative settings in a single page or dialog box. For example, if Standard users can change some but not all settings, split those settings out as a separate UI surface.
- Don't consider the need to elevate when determining if a control should be displayed or disabled. This is because:
  - In unmanaged environments, assume that Standard users could elevate by asking an administrator. Disabling controls that require elevation would prevent users from having administrators elevate.
  - In managed environments, assume that Standard users can't elevate at all. Removing controls that require elevation would prevent users from knowing when to stop looking.

- To eliminate unnecessary elevation: If a task *might* require elevation, elevate as late as possible. If a task needs a confirmation, display the elevation UI only after the user has confirmed. If a task always requires elevation, elevate at its entry point.
- Once elevated, stay elevated until elevated privileges are no longer necessary. Users shouldn't have to elevate multiple times to perform a single task.
- If users must elevate to make a change but choose not to make any changes, leave the positive commit buttons enabled but handle the commit as a cancel. Doing so eliminates users having to elevate just to close a window.
- Don't display an error message when tasks fail because users chose not to elevate. Assume that users intentionally chose not to proceed, so they won't regard this situation as an error.
- Don't display warnings to explain that users might need to elevate their privileges to perform tasks. Let users discover this fact on their own.
- Display the UAC shield and elevation UI based on the following table:


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
833 / 867
100%
Find

Object	Circumstance	Where to put UAC shield	When to elevate
Program	Entire program is for administrators only.	wlsetup-custom Windows Live Installer Microsoft Corporation *UAC shield overlay on program icon.*	Display elevation UI at launch.
Command	Entire command is for administrators only.	Change account type *UAC shield on command button or link.*	Display elevation UI when command button or link is clicked, but after any confirmations.
Command	Command displays useful read-only information appropriate for all users, but changes require administrative privileges.	Computer name, domain, and workgroup settings Computer name: Jonathan-PC Change settings Full computer name: Jonathan-PC.corp.fabrikam.com Computer description: Domain: corp.fabrikam.com *UAC shield on command button or link to make changes.* Screen shot of Change settings link and UAC shield	Display elevation UI when command button is clicked, but after any confirmations.

Figure 3-95: Change settings link and UAC shield


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
833 / 867
100%
Find

Command | Standard users can view the information and possibly make some changes without elevation. Allow Standard users to attempt and | Display elevation UI when user retries the command.

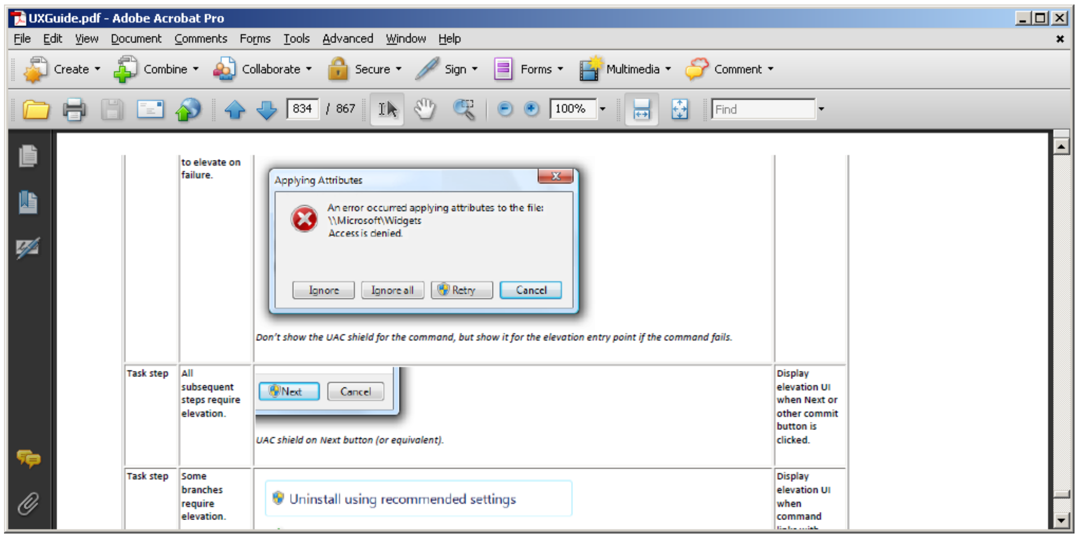
© 2010, Microsoft Corporation. All rights reserved. Page 833 of 867

to elevate on failure.

Applying Attributes

An error occurred applying attributes to the file:
\\Microsoft\Widgets
Access is denied.

Figure 3-96: Applying Attributes warning message


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
834 / 867
100%
Find

to elevate on failure.

Applying Attributes

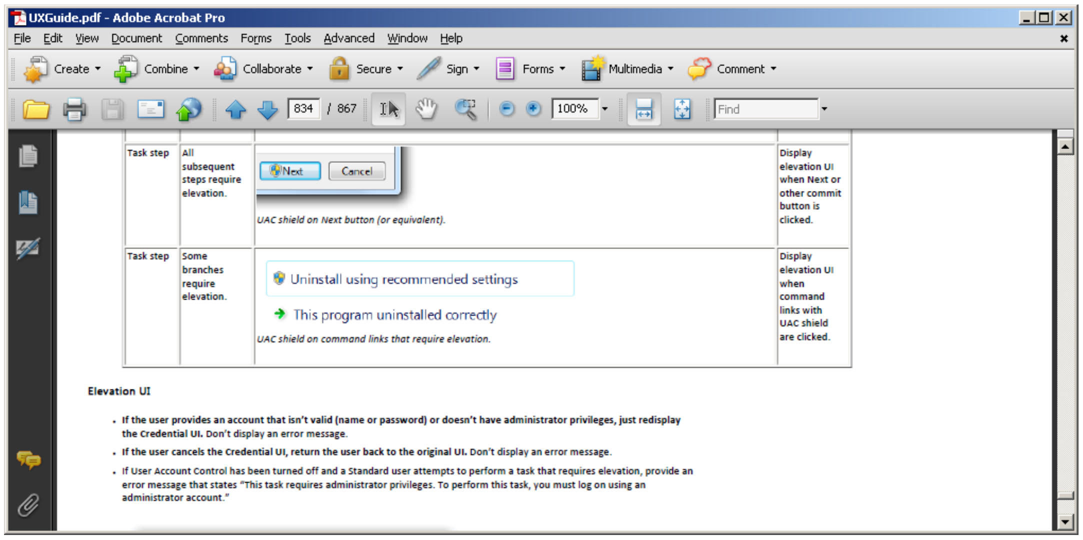
An error occurred applying attributes to the file:
\\Microsoft\Widgets
Access is denied.

Ignore
Ignore all
Retry
Cancel

Don't show the UAC shield for the command, but show it for the elevation entry point if the command fails.

Task step	All subsequent steps require elevation.	Next Cancel *UAC shield on Next button (or equivalent).*	Display elevation UI when Next or other commit button is clicked.
Task step	Some branches require elevation.	Uninstall using recommended settings	Display elevation UI when command

Figure 3-97: Clicking the Next button


UXGuide.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
Create Combine Collaborate Secure Sign Forms Multimedia Comment
834 / 867 100% Find
Next Cancel
Uninstall using recommended settings
This program uninstalled correctly

Task step	All subsequent steps require elevation.	*UAC shield on Next button (or equivalent).*	Display elevation UI when Next or other commit button is clicked.
Task step	Some branches require elevation.	*UAC shield on command links that require elevation.*	Display elevation UI when command links with UAC shield are clicked.

Elevation UI

- If the user provides an account that isn't valid (name or password) or doesn't have administrator privileges, just redisplay the Credential UI. Don't display an error message.
- If the user cancels the Credential UI, return the user back to the original UI. Don't display an error message.
- If User Account Control has been turned off and a Standard user attempts to perform a task that requires elevation, provide an error message that states "This task requires administrator privileges. To perform this task, you must log on using an administrator account."

Figure 3-98: Elevation UI

3.79.3 Elevation UI

- If the user provides an account that isn't valid (name or password) or doesn't have administrator privileges, just redisplay the Credential UI. Don't display an error message.
- If the user cancels the Credential UI, return the user back to the original UI. Don't display an error message.
- If User Account Control has been turned off and a Standard user attempts to perform a task that requires elevation, provide an error message that states "This task requires administrator privileges. To perform this task, you must log on using an administrator account."

3.79.4 Wizards

- **Don't elevate multiple times.** Once a wizard is elevated, it should stay elevated.
- If the task is performed within the wizard, put a UAC shield on the Commit page's "Next" button (which should be given a more **specific label**). When the user commits: If the next page is a Progress page, advance to that page and modally display the elevation UI. After successful elevation, perform the task.
- If the next page is a Completion page, advance to that page (but temporarily replace its contents with "Waiting for permission...") and modally display the elevation UI. After successful elevation, perform the task, and then display the Completion page contents.
- If the user cancels the elevation UI, return to the Commit page. Doing so allows the user to try again.

- If the task is performed after the wizard completes, put a UAC shield on the Commit page's "Finish" button (which should be given a more **specific label**). When the user commits: Remain on the Commit page and modally display the elevation UI. After successful elevation, close the wizard.
- If the user cancels the elevation UI, return to the Commit page. Doing so allows the user to try again.
- For lengthy wizards intended only for administrators, you can prompt for administrator credentials at the entry point before showing any UI.

3.79.5 Text

Don't use an ellipsis just because a command requires elevation. The need to elevate is indicated with the UAC shield.