DISCLAIMER

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Aero Wizard UI Spec

PROGRAM MANAGER	Vinny Pasceri	USABILITY	Joey Benedek
DEVELOPER	Jeff Miller	SDK WRITER	Eric Tilleson
TESTER	Siva Mopati	DEVELOPER LEAD	Jeff Miller
DESIGN	Randy Winjum	TEST LEAD	Paul Trieu
USER ASSISTANCE	Everett McKay	ARCHITECT	Chris Guzak

Figure Reference

Table Reference

1. Overview & Scope

Longhorn will include a set of changes to the existing Win32-based Wizard '97 framework (PSH_WIZARD97) to allow for allow for more interesting visual designs, more usable Task Pages, more effective use of text in wizards, and consistency with changes being made to Explorer. The new framework will be used for new Wizards in Longhorn. The new framework, which continues to be based on Win32, is referred to here as the **Aero Wizard Framework**.

This spec covers changes to the programmatic framework used by developers to create Wizards. It also makes reference to new guidelines being developed for how this framework should be used in practice. The complete set of guidelines for designers and writers working on Aero Wizard wizards are covered separately at the user experience web site.

This spec covers:

- UX Details for the Aero Wizard Framework

2. Definitions

Wizard '97 (internally defined as PSH_WIZARD97) – Wizard '97 is the Wizard Framework that was implemented in 1997 and is used as the primary framework for building wizards from 1997 on.

Aero Wizard (internally defined as PSH_AEROWIZARD) – Aero Wizard is the next generation Wizard Framework that will ship with Longhorn. It will be the framework of choice for building new wizards.

Task Page – A Task Page is an individual page of a wizard. A Task Page is not a certain implementation of a Task Page, a Task Page tells users what to do on the page, and users respond using the page's controls. A Task Page contains a Main Instruction, a body (which can contain text or controls) and Commands.

Non-Client Area – This is a region of a Window which is rendered by the Desktop Window Manager (DWM). The non-client area renders the Caption area and window borders.

Title Bar – This is the area of the Wizard which will display the Back Button, Wizard Icon & Wizard Title.

Header – This is the area of the Wizard which will display the main instruction of a particular Task Page.

Command Area – This is the area of the Wizard which the main content of the Wizard is rendered.

Command Buttons – Command Buttons are the push buttons displayed in the Command Area of the Task Page. This is the default style for Task Pages.

Command Links – Command Links renders the content area of the Wizard with a set of "hyperlinks" which allows the user to navigate to certain Task Pages of the Wizard. The Command Area is not displayed when Command Links are used.

3. Goals

3.1 Problems & Opportunities

The Microsoft Windows user interface model includes a standard interface element called a wizard: a modal dialog that steps the user through some generally linear sequence of pages to accomplish a result. Microsoft helps ISVs create wizards by providing a wizard framework that has two parts:

1. A standard wizard *frame* in the Windows common control library that gives developers programmatic support for wizards. A developer creates pages and then invokes the frame to host these pages. The frame includes standard controls for navigating through the wizard (Back, Next/Finish, and Cancel), a standard region for displaying a watermark graphic, and a standard header region for displaying a header and subhead.
2. A set of user interface *guidelines* aimed at visual designers and user assistance writers producing designs and text for Task Pages. These guidelines dictate conventions for the size of a Task Page, the position and size of typical elements on a page, rules for margins between elements, fonts and sizes, colors, the tone of text on pages, and so on.

The most recent wizard framework is known as Wizard '97. As its name implies, this standard has remained unchanged for a number of years and has not kept pace with advances in computer hardware and user interface design. For example, the Wizard '97 framework dictates the use of a fixed-size modal wizard frame small enough to fit on a VGA screen. ISVs, including Microsoft, are routinely forced to hack around limitations of the Wizard '97 frame and violate the guidelines in order to create a wizard that takes advantage of larger screens to provide a better user experience.

Separately, Microsoft is developing a new application user interface model for tasks that is built on a foundation of web-style navigation. The most common frame for such navigation will be general-purpose Explorer windows. Because navigation is a critical part of the Longhorn user experience, it's important that Explorer and wizards provide consistent controls and behavior for navigation to the degree feasible.

Finally, the Wizard '97 framework was designed for a much older visual style appropriate for older displays and graphics hardware. Microsoft is developing a new visual style for Longhorn called Aero, and it is unpalatable to create new Windows features that don't take advantage of this new style.

3.2 Goals

3.2.1 Make it easier for users to determine the task at hand for each step of the wizard (p1)

3.2.1.1. Reduce text clutter in the header of the wizard (p1)

3.2.1.2. Make the task title more apparent to the user (p1)

3.2.2 Streamline the wizard flow by reducing the number to default steps (p1)

3.2.2.1. Reduce unnecessary pages of the Wizard, such as the Welcome & Finish pages (p1)

3.2.3 Update the Wizard frame visuals to match the Aero Guidelines & Principles (p1)

3.2.3.1. Aero Wizard frame relies on Theme metrics, parts and states to render the look and feel (p1)

3.2.3.2. Aero Wizard frame incorporates Aero guidelines such as padding, font face, and font size into the framework, making it easy for developers to obey Aero guidelines easily (p1)

3.2.3.3. Aero Wizard frame is visually consistent with the Explorer frame (p1)

3.2.3.4. Aero Wizard frame offers beautiful animations & transitions between pages (p2)

3.2.4 Address usability issues of Wizard '97 (p1)

3.2.4.1. Aero Wizard offers resizable pages that take advantage of available screen real estate

3.2.4.2. Aero Wizard appears in the taskbar, preventing the user from losing the Wizard

3.2.5 Enrich the ISV platform (p1)

3.2.5.1. Aero Wizard is easy to deploy (p1)

3.2.5.2. Existing wizards can be easily migrated to Aero Wizard style with minimum developer effort. (p1)

3.2.5.3. Aero Wizard offers “branding” capability for placing a custom banner logo or banner in the Wizard (p2)

3.2.6 Existing high visibility wizards across the Windows client are updated to take advantage of the Aero Wizard style for the Longhorn release. (P1)

3.2.7 Wizard ’97 framework remains intact (p1)

3.2.7.1. All existing applications that use the Wizard ‘97 framework will continue to look and work under Longhorn as they do today. (p1)

3.3 Non Goals

3.3.1 Aero Wizard is available down-level.

4. Plan Summary

4.1 Beta 1 Work Items

4.1.1 General Wizard Framework

4.1.1.1. Wizard appears in the Task Bar.

4.1.1.2. Support for Themes.

4.1.1.3. Support for Classic Visual Style.

4.1.1.4. Support for Resizable Wizards & Non-Resizable Wizards.

4.1.2 Aero Wizard Title Bar

4.1.2.1. “Back” Button moves to the top of the Wizard.

4.1.2.2. “Wizard Icon” is moved next to the Back Button.

4.1.2.3. “Wizard Title” is moved from the non-client area to the Titlebar & is properly themed.

4.1.3 Aero Wizard Header

4.1.3.1. Header background is themed.

4.1.3.2. Top of the Header will have rounded corners when themes are enabled.

4.1.3.3. “Header Background Bitmap” uses a nine-grid layout.

4.1.4 Aero Wizard Content Area

4.1.4.1. “Content Area” background will be pulled from the Theme.

4.1.4.2. “Content Area” controls & text will respond to Themes.

4.1.5 Command Buttons

4.1.5.1. Command Area will have rounded corners when themes are enabled.

4.1.6 Command Links

4.1.6.1. Support for Command Links on a Task Page.

4.1.7 Documentation

4.1.7.1. Aero Wizard API Documentation.

4.2 Beta 2 Work Items

4.2.1 Aero Wizard Title Bar

4.2.1.1. Render the title bar with “Client Glass”

4.2.2 General Wizard Framework

4.2.2.1. Enable smooth frame transition animations when navigating page to page.

5. Window Frame & Header Detailed Design

The section describes the changes and new features of the Aero Wizard framework. **Please note; all pixel values used in this spec are defined at 96 DPI.**

5.1 Requesting the Aero Wizard style

5.1.1 The programmatic interface for Aero Wizard is an extension of that for Wizard '97 to include new options. A developer familiar with creating Wizard '97 wizards will have no problem creating Aero Wizard wizards or porting old wizards to the new style. To ensure that older applications don't unexpectedly receive the new style with potentially confusing results, a developer must explicitly request the Aero Wizard style when instantiating the wizard. This same backward-compatible approach was employed when Wizard '97 was developed (If a developer doesn't specify any wizard style at all, they will end up with an even older style designed for Windows 95 and NT 4.0.).

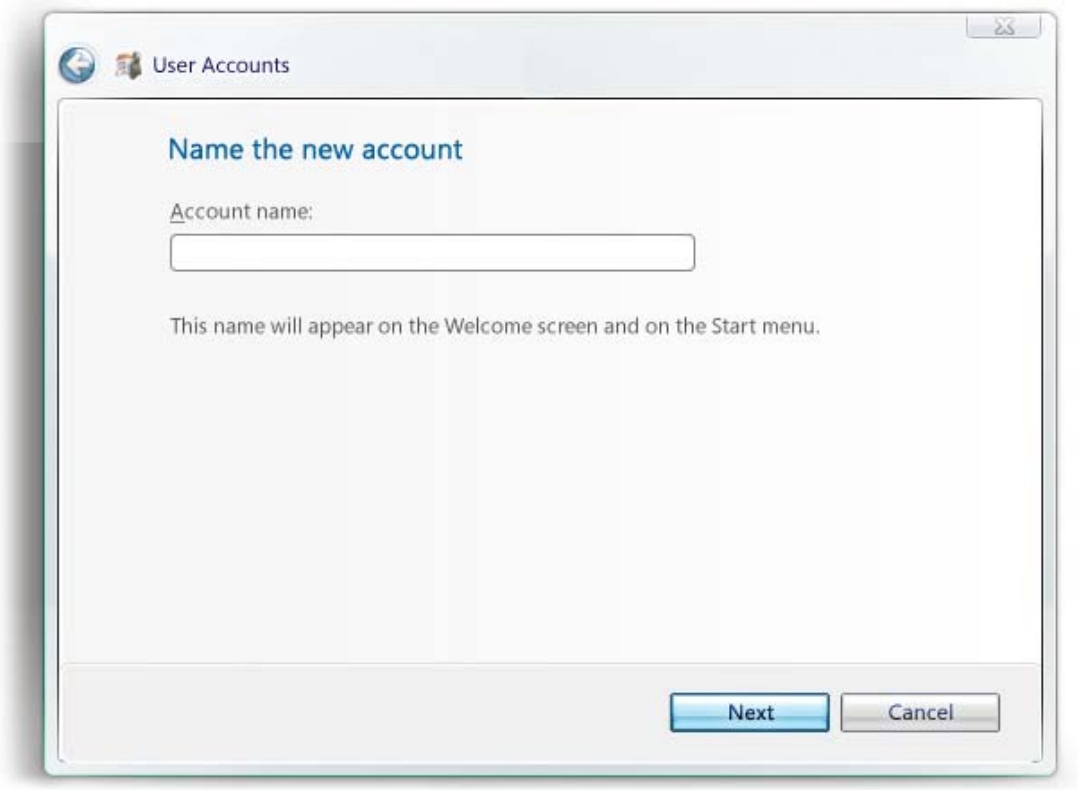


Figure 1 – Standard Aero Wizard Style (using Command Buttons)

5.2 Recommended Sizes & Wizard Resizing

5.2.1 Overview

5.2.1.1. Wizards are just a set of property sheets grouped together and called by the Wizard framework. To make all of the wizards in Windows look similar, we have a set of recommended sizes developers should use.

5.2.1.2. Second, although by default Aero Wizard frame is fixed in size, the frame can optionally allow resizing of the window. It is up to each Task Page to respond to a change in window size. For example, a single scrollable control such as a list box might grow in direct proportion to the page's size. Because Win32 dialogs do not support a sophisticated layout engine, Task Pages will generally resize in relatively simple ways. Specifically, it is not expected that static text controls will re-flow in response to a change in page size, nor will controls below text controls move up or down in response. To compensate for this weakness, the wizard frame does not allow the user to make the window smaller than its initial

size (The window's initial size is determined from the maximum initial height and width of all the pages.)

5.2.2 Building a Wizard

5.2.2.1. If you are building a Wizard using Aero Wizard, we have a recommended size.

5.2.2.2. Default Wizard Size

5.2.2.2.1. Wizards have a **38 pixel (@ 96 DPI)** left margin built into the Framework, which cannot be altered or removed when using a standard wizard style.

5.2.2.2.2. The Content Area should not exceed **317 x 143 dialog units (DLU)** or **555 x 286 pixels (@ 96 DPI using Segoe UI, 9pt).**

5.2.2.3. Custom Wizard Size

5.2.2.3.1. Aero Wizard will support any size wizard you pass in to it.

5.2.2.3.1.1. If the width of height of the Wizard exceeds the desktop size, the Aero Wizard will fall off the screen.

5.2.3 Taskbar Presence

5.2.3.1. Wizards built on the Aero Wizard framework will appear in the Taskbar by default.

5.2.3.1.1. If the wizard has a parent window, it's up to the parent Window to give the Wizard Taskbar presence.

5.2.4 Wizard Sizing Summary

5.2.5 Win32 Task Pages, non-resizable wizard

5.2.5.1. Find the "largest" content width and height of all pages, and use that as the content area size (the frame will "stretch" around that)

5.2.6 Win32 Task Pages, resizable wizard

5.2.6.1. Each page determines the content size. The wizard frame will resize to fit each page.

5.2.7 Wizard Resizing

5.2.7.1. Wizard authors can request a resizable wizard.

5.2.7.1.1. By default wizards are fixed in size.

5.2.7.2. Reflow

5.2.7.2.1. Content section resizes as the frame size changes.

5.2.7.2.2. The Wizard Frame does not allow the user to make the window smaller than its initial size.

5.2.7.2.3. The Aero Wizard Framework does not handle any reflow of text or other elements in the content section of the Wizard if the developer is using Win32. Reflow is only supported in DUI pages.

5.2.7.3. Header area resize

5.2.7.3.1. Horizontally – the header text reflows so that it takes advantage of the entire width of the header before wrapping to the second line.

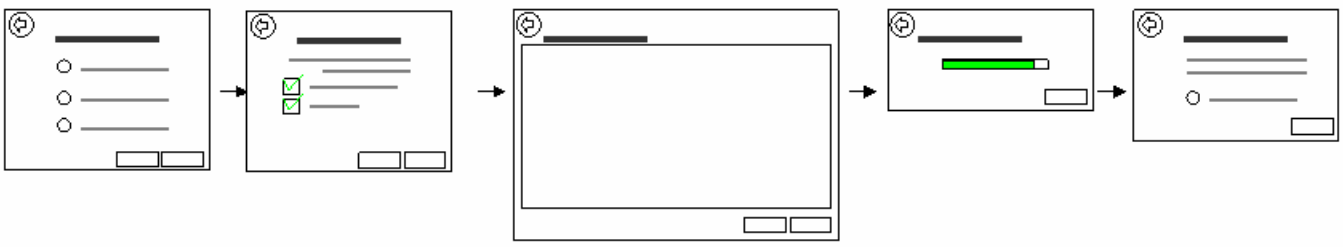


Figure 2 - Auto Resizing the Aero Wizard Frame

5.2.8 Auto Resizing Details

5.2.8.1. "Auto Resizing" will happen automatically when the developer specifies the "Resize" flag for Aero Wizards.

5.2.8.2. The Frame of the Wizard will resize to the content area of the Wizard Property Page.

5.2.8.3. If the Developer is using a Win32 Property Sheet, the Wizard will resize to fit to the size of the specified Win32 Property Sheet.

5.2.8.4. If the developer leaves blank space in the Win32 Property Sheet, the Wizard Frame will consider all white space to be content thus not resizing to a more appropriate size.

5.2.8.4.1. **Figure 3** shows an example of a Win32 Property Sheet will white space.

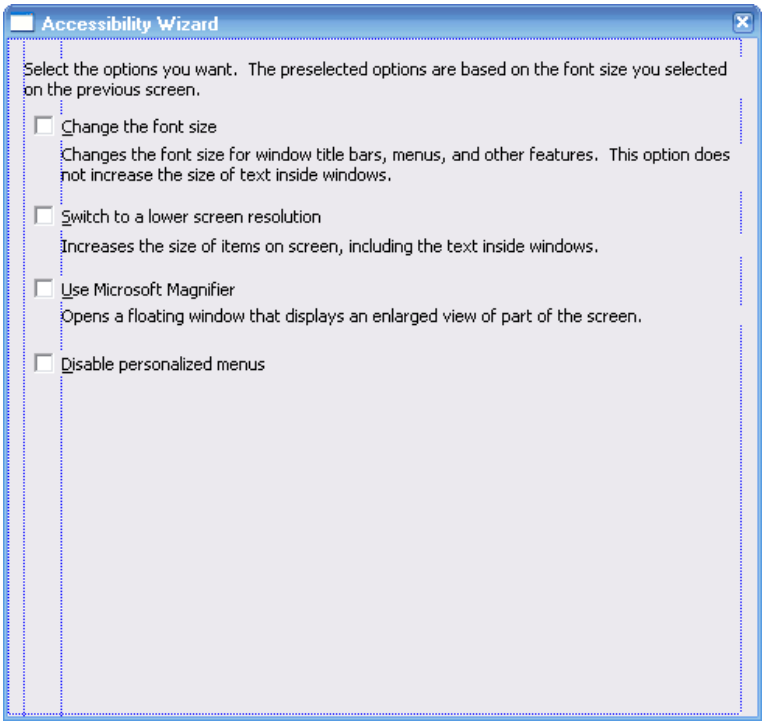


Figure 3 - A Win32 Property Sheet with Blank Content (BAD!)

5.2.8.5. The developer must manually size the Property Sheet to the "size of the content" if they want to take advantage of this feature.

5.2.8.5.1. **Figure 4** shows an example where the Developer "sized to content" in a Win32 Property Sheet. All property sheets of the wizard must be modified to "size to content" to use the Auto Resize feature.

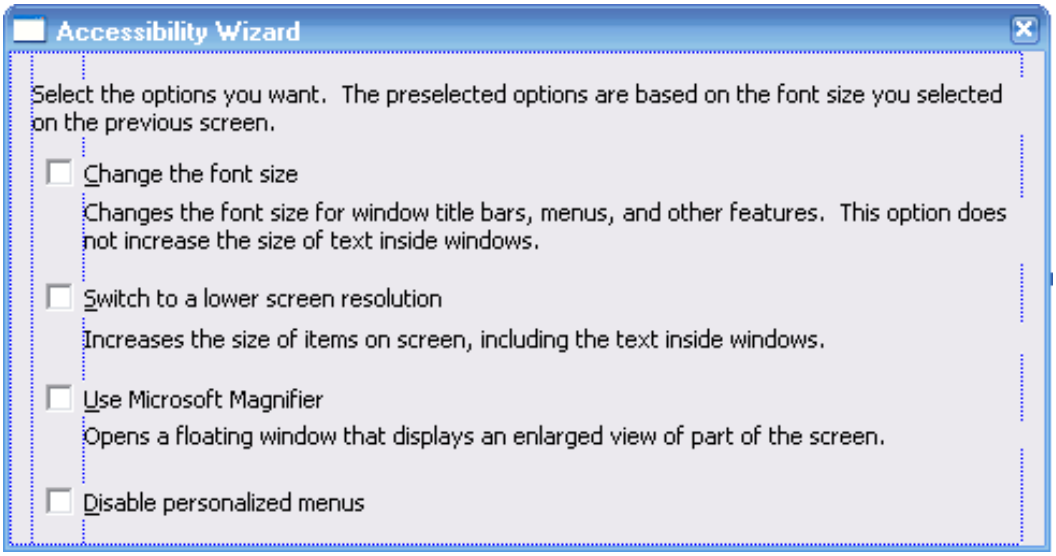


Figure 4 - A Win32 Property Sheet Sized to Content (GOOD!)

5.2.8.6. If the Developer is using a DUI Page, the Wizard will resize appropriately to fit to the size of the specified DUI Page.

5.2.8.7. As a guideline, we would like to resize the Wizard vertically, not horizontally to prevent the command buttons from jumping around too much.

5.2.8.7.1. This is not enforced by the Framework, as the Wizard developer is responsible for ensuring each Property Sheet has the same width.

5.2.9 Aero Wizard Placement on the Desktop

5.2.9.1. When created, the Wizard will be placed in the center of the parent Window.

5.2.9.1.1. The developer can choose to move the placement of the Wizard when created.

5.2.9.2. The Wizard will retain the (X,Y) coordinates for the upper left hand corner of the Wizard when the wizard is resized for each page.

5.2.9.3. If the user moves the Wizard, the new (X,Y) coordinates will be used when navigating to the next page, it will not revert to the (X,Y) coordinates when it was first opened.

5.2.9.4. The new (X,Y) coordinates will not be saved when the user is done with the Wizard.

5.2.10 Maximized Wizards

5.2.10.1. If the user maximizes the Wizard, the rest of the Wizard will be maximized, it will not resize.

5.2.11 Manual Resizing & Saving State

5.2.11.1. If the wizard is manually resized, the command buttons are pegged to the lower right corner of the Wizard.

5.2.11.2. If a user resizes the Wizard on any page, goes to the next page and then returns back to the resized page, the Wizard will save the state of the Resized page.

Table 1 - General Framework: Summary of Changes

Component	Summary of Changes from Wizard '97
	Wizards will show up in the Task Bar.
	Aero Wizard offers a flag for resizing the Wizard.
	Add support to resize the Frame to the size of the Property Page.
	Add support to keep the placement of the wizard when the user advances through the wizard.
	Do not resize the Wizard when the Wizard is Maximized.
	Resize -> Next -> Back – Save the state of the resized page.
	Command Buttons are pegged to the lower right corner on resize.

5.3 Aero Wizard Components

5.3.1 Aero Wizard is broken into four main UI components:

5.3.1.1. Title bar

5.3.1.2. Header

5.3.1.3. Content Region

5.3.1.4. Command Area

5.3.2 The Title bar, Header, and Command Area components are all drawn by the framework. No additional work is needed for developers to get these areas to look like Aero.

5.3.3 The Content Area inside the Content Region of Aero Wizard is where the developer inserts Task Pages for their wizard.

5.3.4 **Figure 5** below displays all of the major UI components for Aero Wizard in a conceptual diagram.

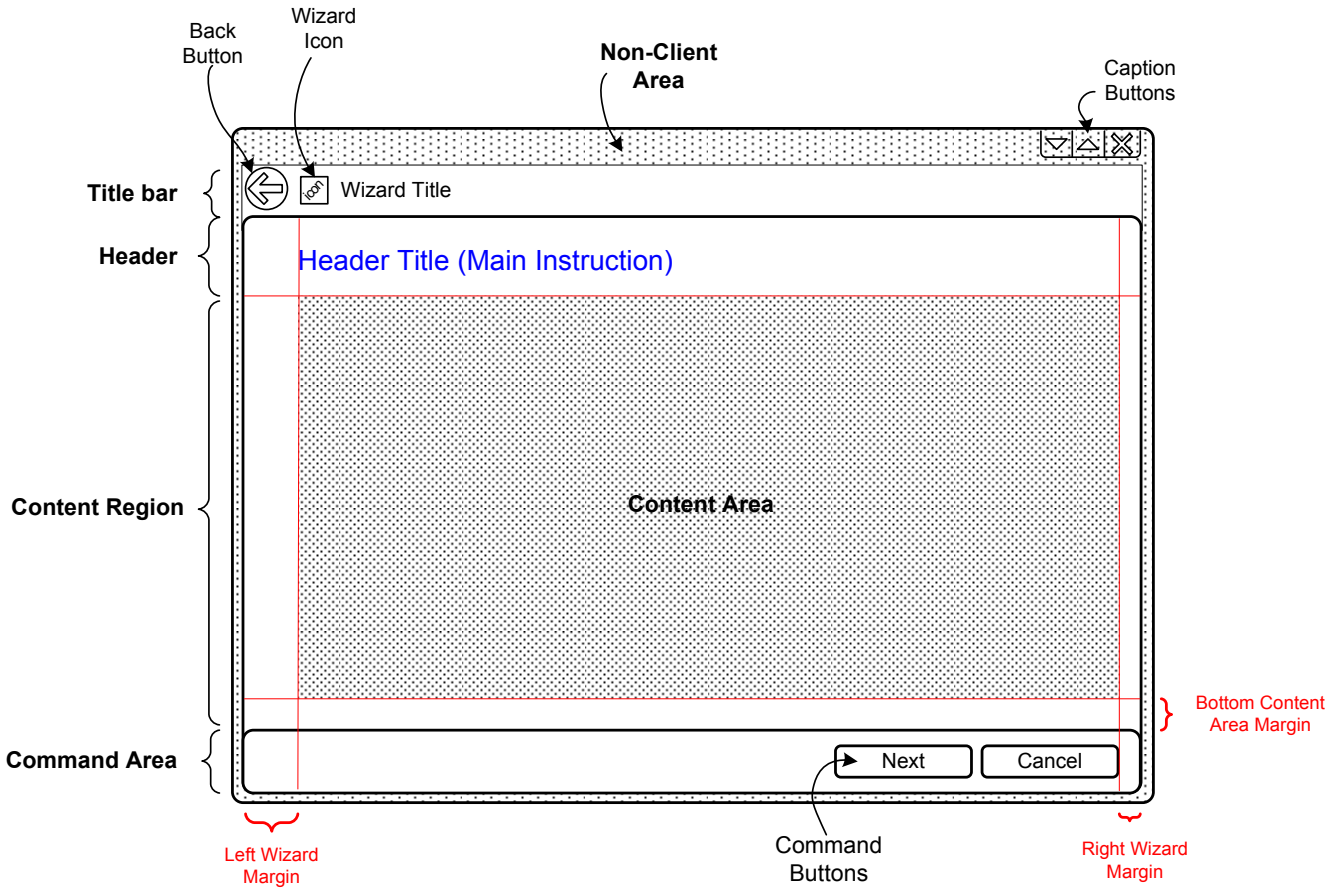


Figure 5 - Aero Wizard w/Command Buttons Conceptual Diagram

5.4 Wizard Frame Visuals

5.4.1 The Frame consists of the Title bar, Caption Buttons and Window Border.

5.4.2 The Aero Wizard Frame should be visually consistent with the Explorer Frame.

5.4.2.1. If the Explorer Frame can support glass in the client area, the Aero Wizard Framework should as well in the Title Bar.

5.5 Title Bar Overview

5.5.1 The Wizard Title Bar is composed of the following components:

5.5.1.1. Back Button

5.5.1.2. Wizard Icon

5.5.1.3. Wizard Title

5.5.2 **Figure 6** is the conceptual diagram for the Aero Wizard Title Bar.

5.5.3 **Figure 7** is the Visual Spec for the Aero Wizard Title bar which specifies all margins and dimensions.

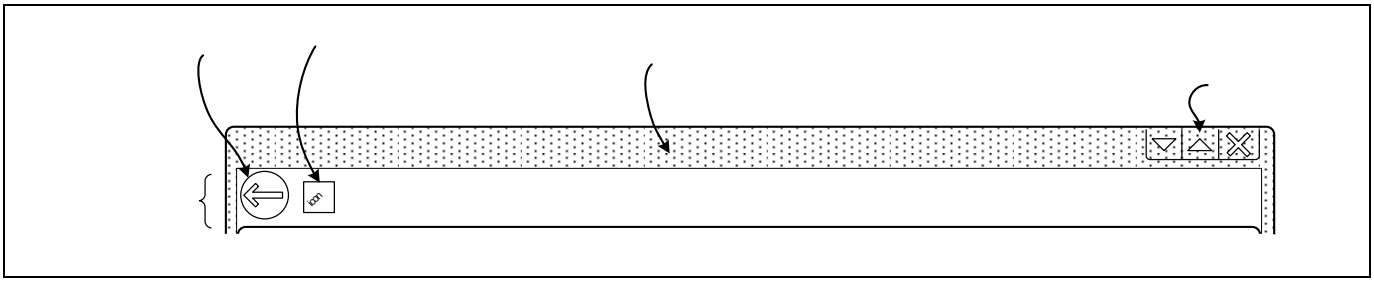


Figure 6 - Title Bar Conceptual Model

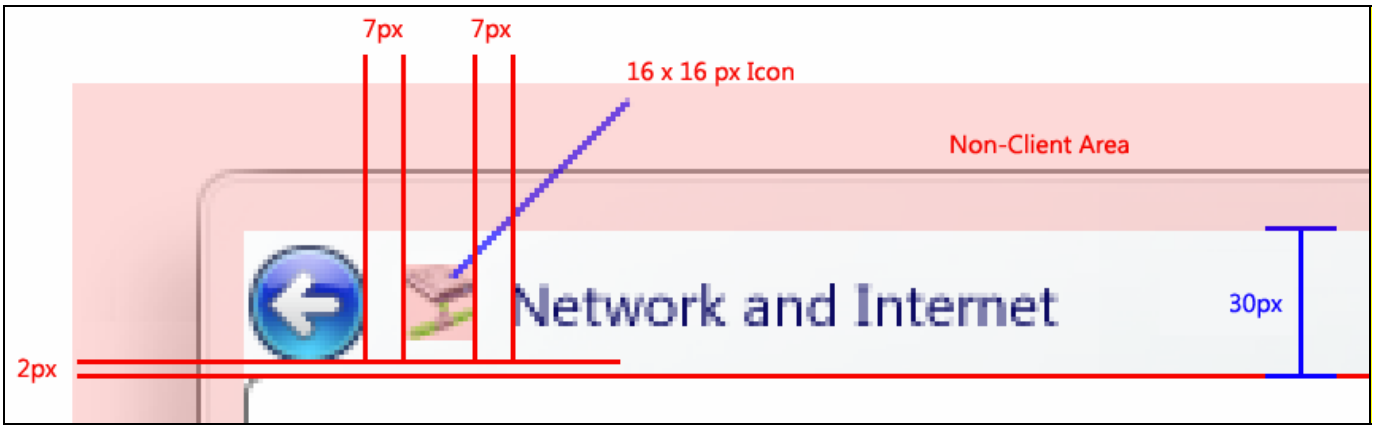


Figure 7 - Aero Wizard Title Bar Visual Spec

Back Button

Wizard Icon

Non-Cl

Title bar

Wizard Title

5.6 Title Bar - Back Button

5.6.1 General

5.6.1.1. A key design element of Explorer is the Back button in the window "chrome" or title area. For consistency with this key Explorer element, the Aero Wizard frame also places the Back button in the window chrome instead of in its old location along the lower edge of the window. The Back button behaves the same as the Back button in Wizard '97. It is simply in a new location, and individual Task Pages are unaware of this change.

5.6.2 Visuals

5.6.2.1. The Back Button always appears in the upper left corner of the Wizard. Developers cannot change the placement of the Back Button.

5.6.2.1.1. For the RTL case, the back button will appear in the upper right corner of the wizard.

5.6.2.2. The size of the Back button is always the size specified by the Theme. Developers cannot change the size of the Back button.

5.6.2.3. The Back Button visuals are specified by the current Theme File.

5.6.2.4. When High DPI is enabled, we will use the appropriate Theme parts for the set DPI value.

5.6.3 Behavior

5.6.3.1. The back button will **not** maintain a history stack of where the user has been.

5.6.3.2. While the Aero Wizard frame does sport an Explorer-style Back button in its window chrome, it does not offer a split button to look at the history.

5.6.3.3. Aero Wizard supports the following keyboard shortcuts for the back button:

5.6.3.3.1. Alt + Left Arrow

5.6.3.3.2. Tabbed to the Back button and press "ENTER"

5.6.3.4. The Back Button will be in a disabled state for the initial page of the Wizard.

5.6.3.5. Developers can choose to hide or disable the back button.

5.6.3.5.1. Developers can set this at any time in the Wizard.

5.7 Title Bar – Wizard Icon

5.7.1 Wizard Icon appears to the left of the Back button as specified by the Visual Spec. Developers cannot change the placement of the Wizard Icon.

5.7.2 Wizard Icon is set per Wizard, not per Task Page.

5.7.3 Wizard Icon uses SM_CXSMICON & SM_CYSMICON for the icon size (**16 x 16 pixels** @ 96 DPI).

5.7.4 Wizard Icon will use the appropriate size when a higher DPI value is used.

5.7.5 Developers can specify a Wizard Icon. If no Icon is specified, no icon will be displayed and the Wizard Title Text will move over.

5.7.6 Wizard Icon will be displayed in the Taskbar as well.

5.7.6.1. If no Wizard Icon is specified, no Wizard Icon will appear in the Taskbar.

5.8 Title Bar - Wizard Title

5.8.1 The Wizard Title always appears in next to the Wizard Icon in the Header. Developers cannot change the placement of the Wizard Title.

5.8.2 The size & style of the Wizard Title are specified by the current Theme File. Developers cannot change the size & style.

5.8.3 The Wizard Title is set per Wizard, not per Task Page.

5.9 Title Bar - Caption Buttons

5.9.1 Caption Buttons appear in the same area of the Header as it is for standard windows.

5.9.2 Minimize, Maximize and close caption buttons appear for resizable wizards.

5.9.3 Non-resizable wizards support only the close caption button.

Table 2 - Aero Wizard Title Bar: Summary of Changes

Component	Summary of Changes from Wizard '97
Aero Wizard Title Bar	"Back" button of the Wizard Framework is moved to the top of the Wizard.
	"Wizard Icon" is displayed in the bar next to the back button.
	The style of the "Wizard Title" is specified by the Theme File.
	"Wizard Title" appears in the client area title bar, not the non-client area.

5.10 Header Overview

5.10.1 Overview

5.10.1.1. There are a variety of changes we are making to the Wizard Header. This section describes the header for Wizard '97 so you can compare the changes to Aero Wizard.

5.10.1.2. Legacy Wizards wishing to migrate to Aero Wizard can still specify all the parameters from Wizard '97, but only the Header Title will be displayed.

5.11 Aero Wizard Header

5.11.1 Overview

5.11.1.1. This section describes the improvements made to the Aero Wizard Header Area.

5.11.1.2. To provide a clean UI look & feel, we removed the Header Sub-Title.

5.11.1.2.1. **Figure 8** shows the conceptual diagram for the Aero Wizard Header.

5.11.1.3. The new Header consists of a Header Title & an optional Header Background Bitmap.

5.11.1.3.1. **Figure 9** shows the old Header from Wizard '97.

5.11.1.3.2. **Figure 10** shows the new Header for Aero Wizard.

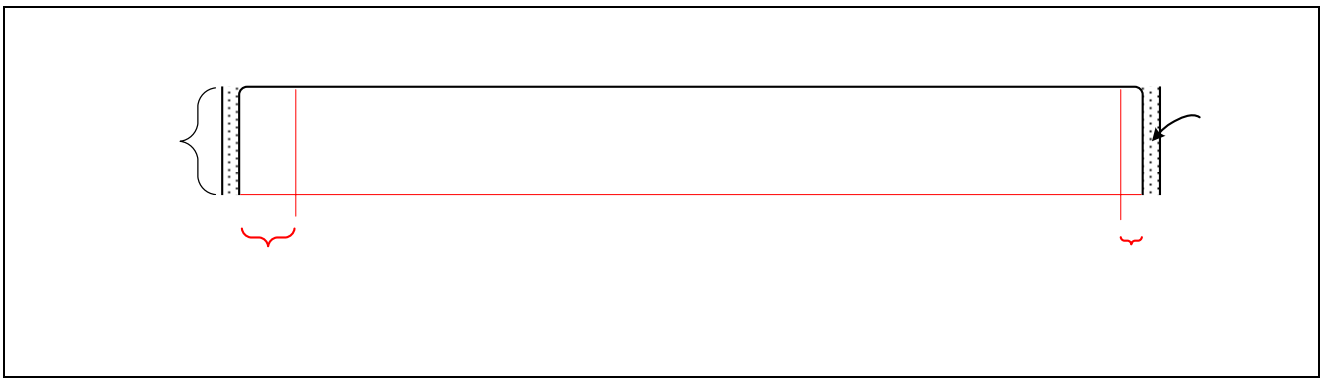


Figure 8 - Header Conceptual Diagram

Figure 9 - Wizard '97 Header

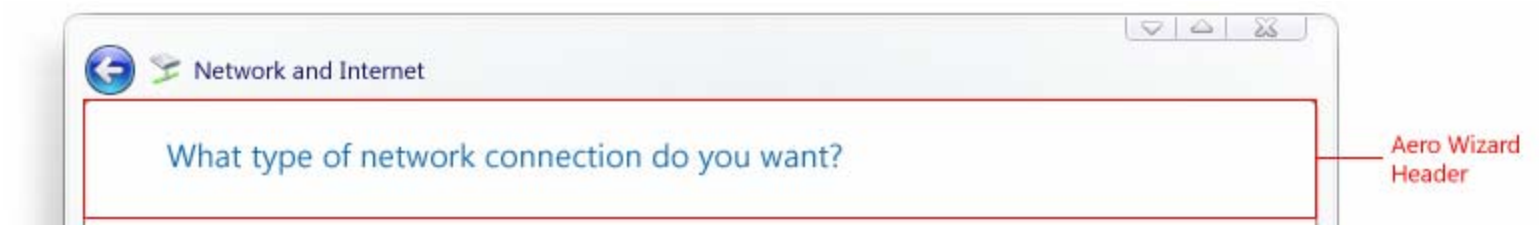


Figure 10 - Aero Wizard Header

5.11.2 Aero Wizard Header Margins

5.11.2.1. There is a left margin which is built into the Wizard framework. This margin affects the header area.

5.11.2.1.1. Even if the developer specifies “No Margin” for the Wizard, we will continue to use the standard margin values for the header.

5.11.2.2. **Figure 11** shows the Aero Wizard Header Visual Spec for small Task Pages which uses a **38 pixel (@ 96 DPI)** left margin.

5.11.2.3. There is also a top and bottom margin for the header text of **19 pixels (@ 96 DPI).**

5.11.2.4. All margin values are stored in the Theme file.

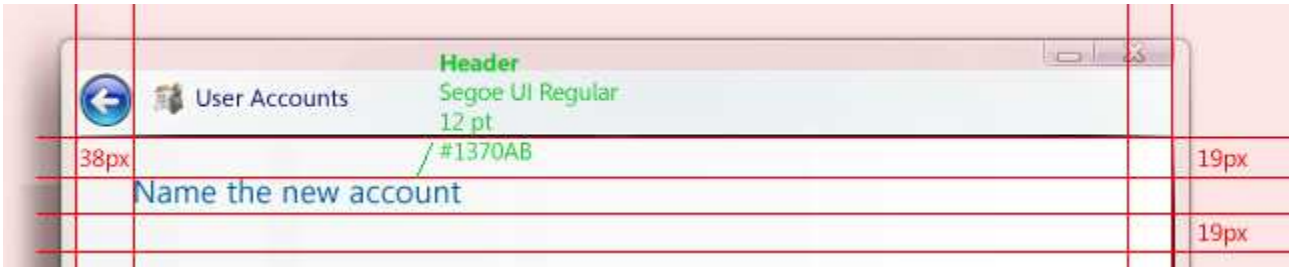


Figure 11 - Small Wizard Header

5.11.3 Header Details

5.11.4 General Header

5.11.4.1. The Top of the Header should be rendered with Rounded corners **(3 pixel radius @ 96 DPI)** when themes are enabled. This value should be pulled from the Theme file.

5.11.4.2. The Header background should be pulled from the Theme file.

5.11.4.3. In Classic Visual Style, the background color will be pulled from system metric values and no rounded corners should appear.

5.11.5 Header Title (Main Instruction)

5.11.5.1. This is the Title of the Task Page.

5.11.5.2. Header Title is defined for each Task Page.

5.11.5.3. The Font Style & Size of the Header Title is specified by the current Theme File.

5.11.5.4. The Header Title can wrap over multiple lines (**Figure 12**).

5.11.5.4.1. Default minimum height is the size of one line of text with the current Theme font.

5.11.5.4.2. Title will re-flow if the Wizard is resized.

5.11.5.4.3. Even if the Header Title flows over multiple lines, the bottom margin **(19 pixels @ 96 DPI)** will remain.

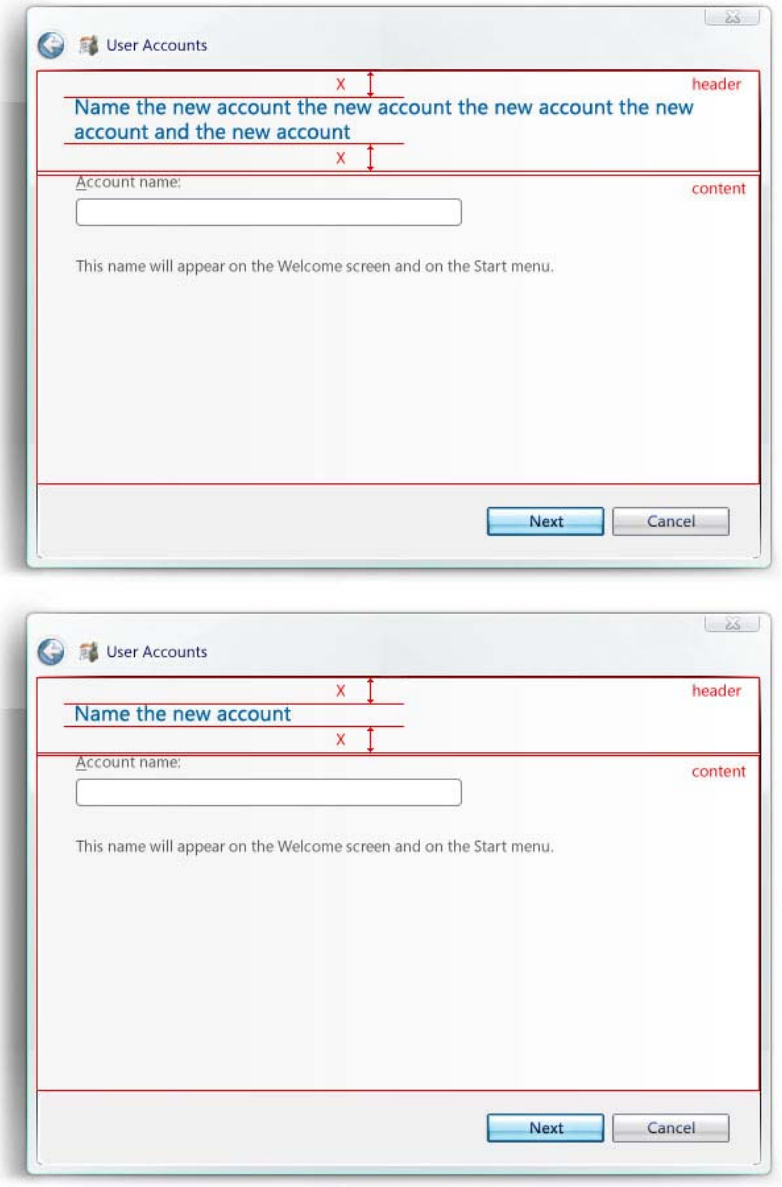


Figure 12 - Flowing Header Title

5.11.6 Header Background Bitmap

5.11.6.1. The developer can choose to define a background bitmap for the header (**Figure 13**).

5.11.6.2. The background bitmap can be any size or any dimension.

5.11.6.2.1. The header area will grow larger to support the bitmap background if needed.

5.11.6.3. A Header Background Bitmap can be defined, but the developer should follow the UX Guidelines on when they should use this.

5.11.6.4. When SPI_HIGHCONTRAST is enabled, Aero Wizard will render the background using system colors instead.

Figure 13 - Header w/Bitmap Background

5.11.6.5. When the developer specifies a Header Background Bitmap, it will be used as a nine-grid (**Figure 14**).

5.11.6.5.1. Section 1 – the upper left pixel will stretch to fill the top and left.

5.11.6.5.2. Section 2 – the top row of pixels will stretch to fill the top.

5.11.6.5.3. Section 3 – the upper right pixel will stretch to fill the top and right.

5.11.6.5.4. Section 5 – this area is static, so this is where any logo should go.

5.11.6.5.5. Section 6 – this column of pixels will stretch to fill the right.

5.11.6.5.6. Section 7 – the column of pixels will stretch to fill the left.

5.11.6.5.7. Section 8 – this row of pixels will stretch to fill the bottom.

5.11.6.5.8. Section 9 – this lower right pixel will stretch to fill the right and bottom.

5.11.6.6. The Header background bitmap will appear to the left of the right Wizard margin (currently at 22px @ 96 DPI).

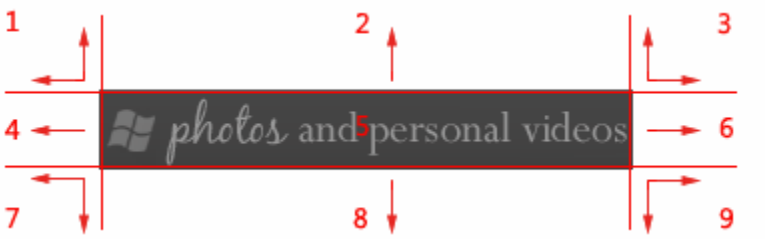


Figure 14 - Header Background Bitmap Nine-grid spec

5.11.6.7. **Figure 15** below shows the Header Bitmap Background in action as well as showing the nine-grid parts.

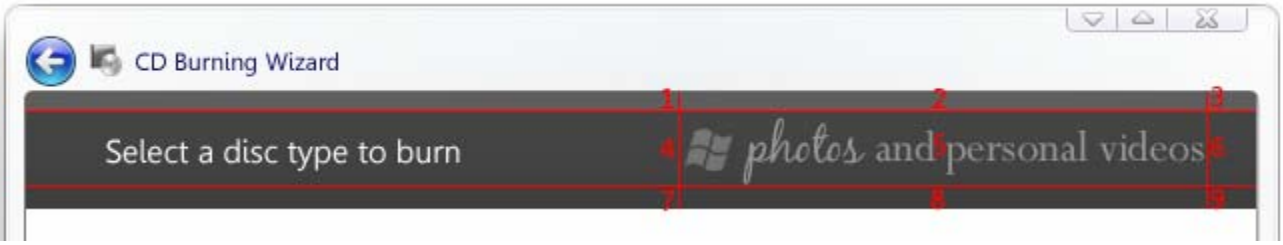


Figure 15 - Header Background Bitmap Example

5.11.7 Setting the Header Title Color

5.11.7.1. When the developer specifies a header background bitmap, the developer can specify a color for the Header Title text.

5.11.7.2. The developer can only change the font color, not the size, style or font name.

5.11.7.3. When SPI_HIGHCONTRAST is enabled, the customized color will revert to standard system colors.

Table 3 - Aero Wizard Header: Summary of Changes

Component	Summary of Changes from Wizard '97
Aero Wizard Header	Header background color should be pulled from the Theme file.
	The Top of the Header should be rendered with rounded corners when themes are enabled. This should be pulled from the Theme file.
	"Header Title" is themed. The style & size is specified by the Theme File.
	"Header Sub-Title" is no longer displayed.
	A Header Background bitmap can be defined as a nine-grid.
	Developer can specify a custom color for the Header text when a Header Bitmap is defined.
	A top and bottom margin will be implemented for the header.
	"Header Title" can wrap over multiple lines.

6. Content Area & Command Area Detailed Design

6.1 Aero Wizard Content Area

For all previous versions of the Wizard Framework (Wizard & Wizard 97), the static text & background color was set to standard system colors.

6.1.1 Overview

6.1.1.1. Content Area of the Aero Wizard Framework is where the developer will place the content for the individual Task Page.

6.1.1.2. Content Area is a Task Page with a set of content (text, controls, etc) hosted in the Aero Wizard Framework.

6.1.1.3. The Content Region is the area in the wizard which contains all viewable area between the Header & Command Area.

6.1.1.4. The Content Area is the region inside the Content Region minus the left, right, top and bottom margins. This is where the Wizard Content is placed (**Figure 16**).

Figure 16 - Content Area Conceptual Diagram

6.1.2 Content Area Dimensions

6.1.2.1. The recommended content area size is **317 x 143 DLUs** or 555 x 286 pixels using Segoe UI, 9pt Font (Figure 17).

6.1.2.1.1. To receive accurate pixel dimensions, it is important to use Segoe UI, 9pt as the font for the content area. DLUs are calculated based on the font & font size specified.

6.1.2.2. Developers can use any size they wish, but we recommend the standard size for consistency across wizards.

6.1.2.3. Developer also has the option to not use the content area left & right margins (**Figure 18**).

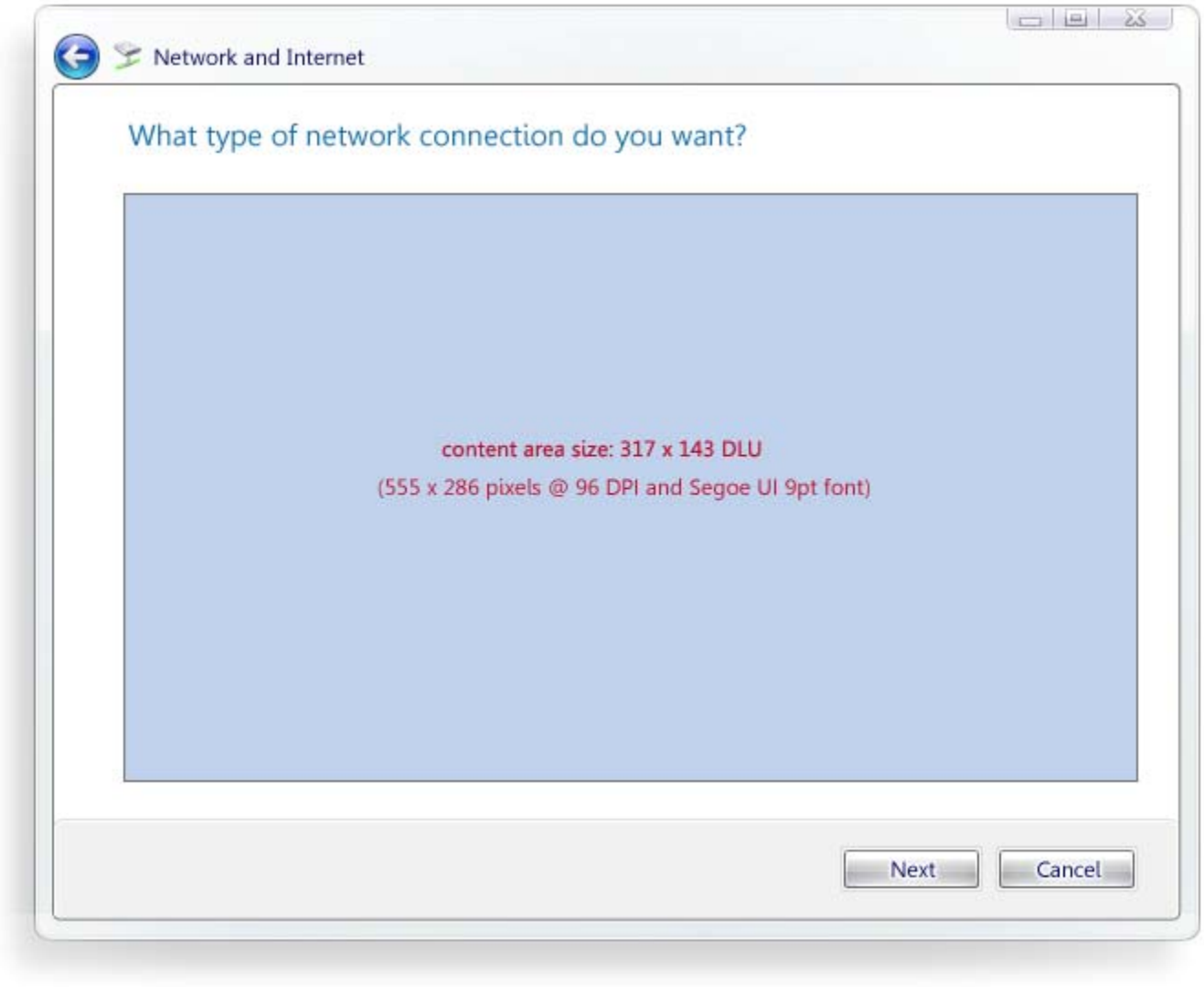


Figure 17 - Small Wizard Template

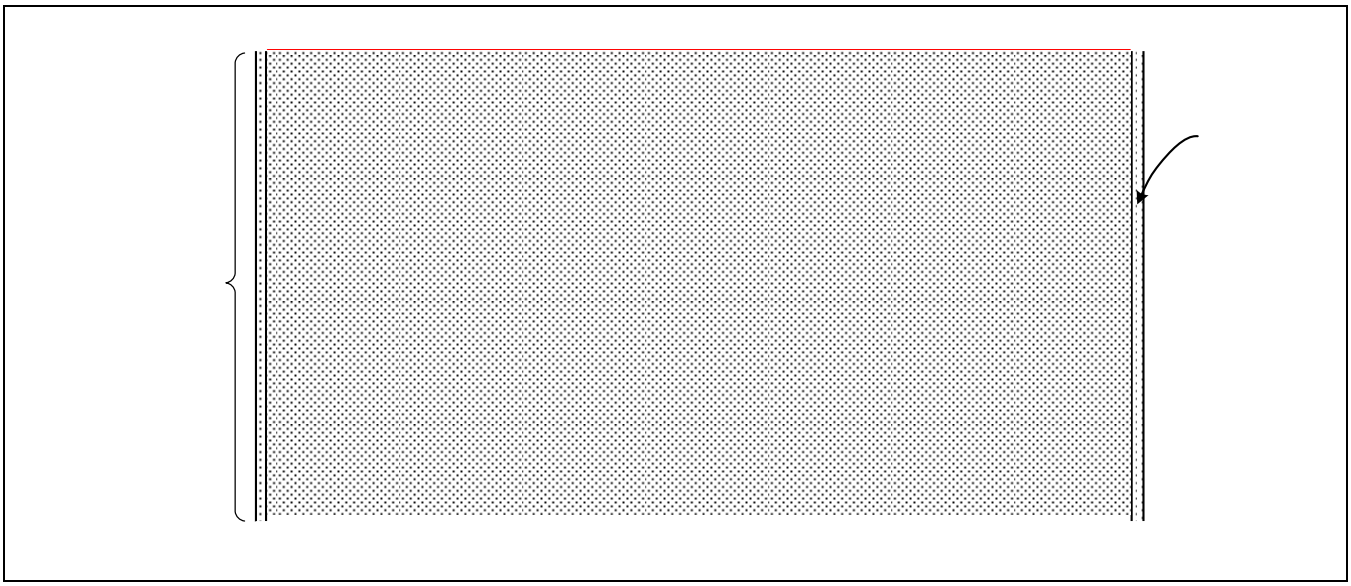


Figure 18 - Content Area Conceptual Diagram - No Margin

6.1.3 Visuals

6.1.3.1. Background

6.1.3.1.1. For Classic Visual Style (non-themed), the Property Sheet background color will be specified from a system metric.

6.1.3.1.2. When Themes are enabled, the background will be pulled from the Theme file.

6.1.3.2. Controls

6.1.3.2.1. For Classic Visual Style (non-themed), the controls rendered in the Content Area will render as classic (non-themed) controls.

6.1.3.2.2. When Themes are enabled, the controls will render with their Themes functionality. This means all states & theme parts will show for the controls.

6.1.3.3. Static Text

6.1.3.3.1. For Classic Visual Style (non-themed), text rendered in the Content Area will render with the Classic Visual Style system font.

6.1.3.3.2. When Themes are enabled, the text will pull the size & style of the text from the current Theme file.

6.1.4 Behavior

6.1.4.1. Developers can specify whatever content they want in the Content Area (text, controls, etc).

6.1.4.1.1. Although developers can specify whatever they want in the Content Area, they should obey the Aero Wizard UX Guidelines.

Table 4 – Aero Wizard Content Area: Summary of Changes

Component	Summary of Changes from Wizard '97
Aero Wizard Content Area	"Content Area" background will respond to a Theme change. For Aero, the background will be white.
	"Content Area" controls & text will respond to Themes. If static text is present, it will get the current font style & size from the current Theme File.
	Developers can use any size content area they wish.
	Developer can specify "No Margin" for the content area.

6.2 Command Area - Buttons

Aero Wizard will have some major improvements in the Command Area which will very useful for developers such as support for button renaming & hiding buttons.

6.2.1 Command Area General

6.2.1.1. The Command Area is rendered at the bottom of the Wizard (**Figure 19**).

6.2.1.1.1. See **Figure 20** for a conceptual diagram.

6.2.1.2. The Command Area will render with rounded corners (**3 pixel radius**) on all sides.

6.2.1.2.1. The Rounded corners metrics will be pulled from the Theme File.

6.2.1.2.2. In Classic Visual Style (non-themed), rounded corners will not appear.

6.2.1.3. The Command Area background will be pulled from the theme file.

6.2.1.3.1. In Classic Visual Style (non-themed), the background will be the standard system color

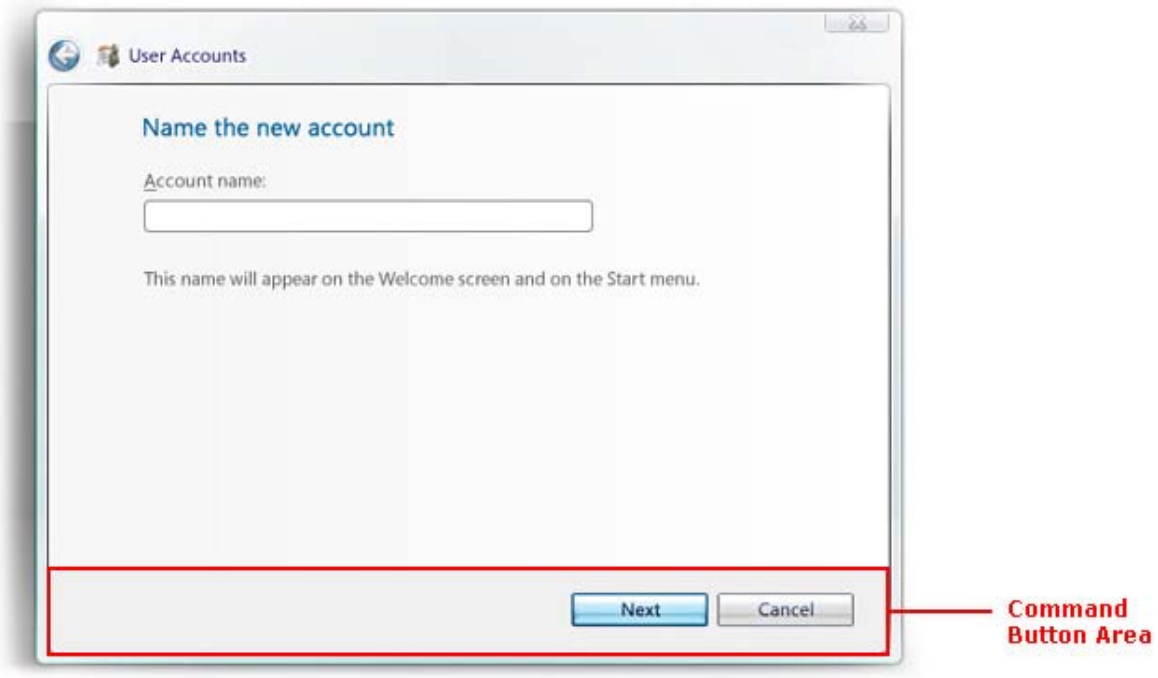


Figure 19 – Aero Wizard Command Area w/Command Buttons

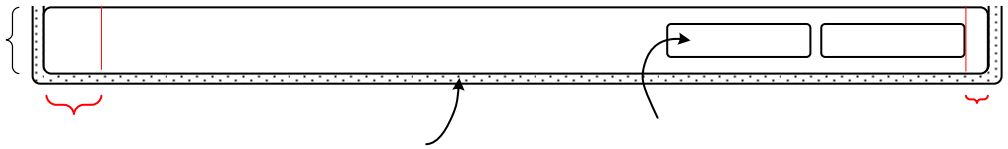


Figure 20 - Command Area Conceptual Diagram

6.2.2 "Next" Button Guidelines & Changes

6.2.2.1. The Aero Wizard frame allows a developer to specify the text that appears on the Next or Finish buttons. This is done to support a new type of "commit" page defined in the Aero Wizard guidelines. A commit page appears before beginning a time-consuming or otherwise significant operation—often the core operation of the wizard. Instead of simply labeling the button that will initiate this operation

"Next", the developer can give more specific text: "Print Pictures", "Send Now", "Start Backup", etc. The button changes its size to accommodate its label.

6.2.2.2. The Aero Wizard frame also supports pages that don't need a Next button at all. In lieu of Wizard '97 pages that asked the user to make a choice using radio buttons and a Next button, the Aero Wizard guidelines recommend letting the user make such a choice by clicking a hyperlink that simultaneously indicates their choice and moves to the next step. This hyperlink can be the standard hyperlink control or the Command Links control.

6.2.3 "Cancel" Button Guidelines & Changes

6.2.3.1. The Aero Wizard frame allows a developer to hide the Cancel button on a page. The Aero Wizard guidelines suggest that Cancel buttons be used on commit pages (see "Next button" above) so that the user has a very obvious way to cancel the operation. Cancel buttons may also be useful on pages in very long wizards. Otherwise, on a page that is simply collecting information from the user, or for pages that appear after a wizard has performed its work, the guidelines recommend hiding the Cancel button.

6.2.4 Hiding buttons

6.2.4.1. Developer can choose to hide the "Next", "Finish", "Cancel" or Back buttons if they are not needed for any Task Page.

6.2.4.2. If all of the Buttons are hidden in the Command Area, the Command Area will be hidden as well.

6.2.5 Renaming the "Next" Button

6.2.5.1. If the button is not renamed, "Next" or a localized variant will be used as the text.

6.2.5.2. The button resizes & repositions to accommodate the text.

6.2.5.3. The **recommended** maximum length of the "Next" Button text is **12 characters.**

6.2.5.3.1. Only Unicode Text is supported.

6.2.6 Renaming the "Finished" Button

6.2.6.1. If the button is not renamed, "Finished" or localized variant will be used as the text.

6.2.6.2. The button resizes & repositions to accommodate the test.

6.2.6.3. The **recommended** maximum length of the "Finish" Button text is **10 characters**.

6.2.6.3.1. Only Unicode Text is supported.

6.2.7 Button Sizing Behavior

6.2.7.1. All of the buttons used in the Command Area will be normalized to a point: When the delta between the custom string width and "Cancel" goes above 15%, only the custom button gets wider from that point on. This will prevent widths that are just slightly different – avoiding a visual bug.

Table 5 - Command Area: Summary of Changes

Component	Summary of Changes from Wizard '97
Command Area	The Command Area should render rounded corners on all sides of the Command Area. This should only appear when Themes are enabled.
	"Next" & "Finish" Command Buttons can be renamed.
	"Next", "Finish", "Cancel", and Back buttons can be hidden.
	If all buttons are hidden, the command area is hidden.
	Button size normalization.

6.3 Command Link Pages

6.3.1 Overview

6.3.1.1. For an alternative to using Command Buttons for a Task Page, developers can use **Command Links** (Figure 21).

6.3.1.2. A Command Link is a visually complex Push Button.

6.3.2 Visuals

6.3.2.1. A Command Link typically consists of a default glyph, header and sub-header supporting text.

6.3.2.2. A developer can specify a different glyph to be used (e.g. an Icon).

6.3.3 Behavior

6.3.3.1. Command Links are the primary way a user navigates to the next Task Page.

6.3.3.2. To make it easy for users to exit out of a Task Page with Command Links, we will continue to display the Command Area and render a “Cancel” button.

6.3.3.2.1. Developers will still need to specify what buttons are displayed for Task Pages with Command Links.

6.3.3.2.2. Developers still have the option to remove any and all buttons when using Command Links.

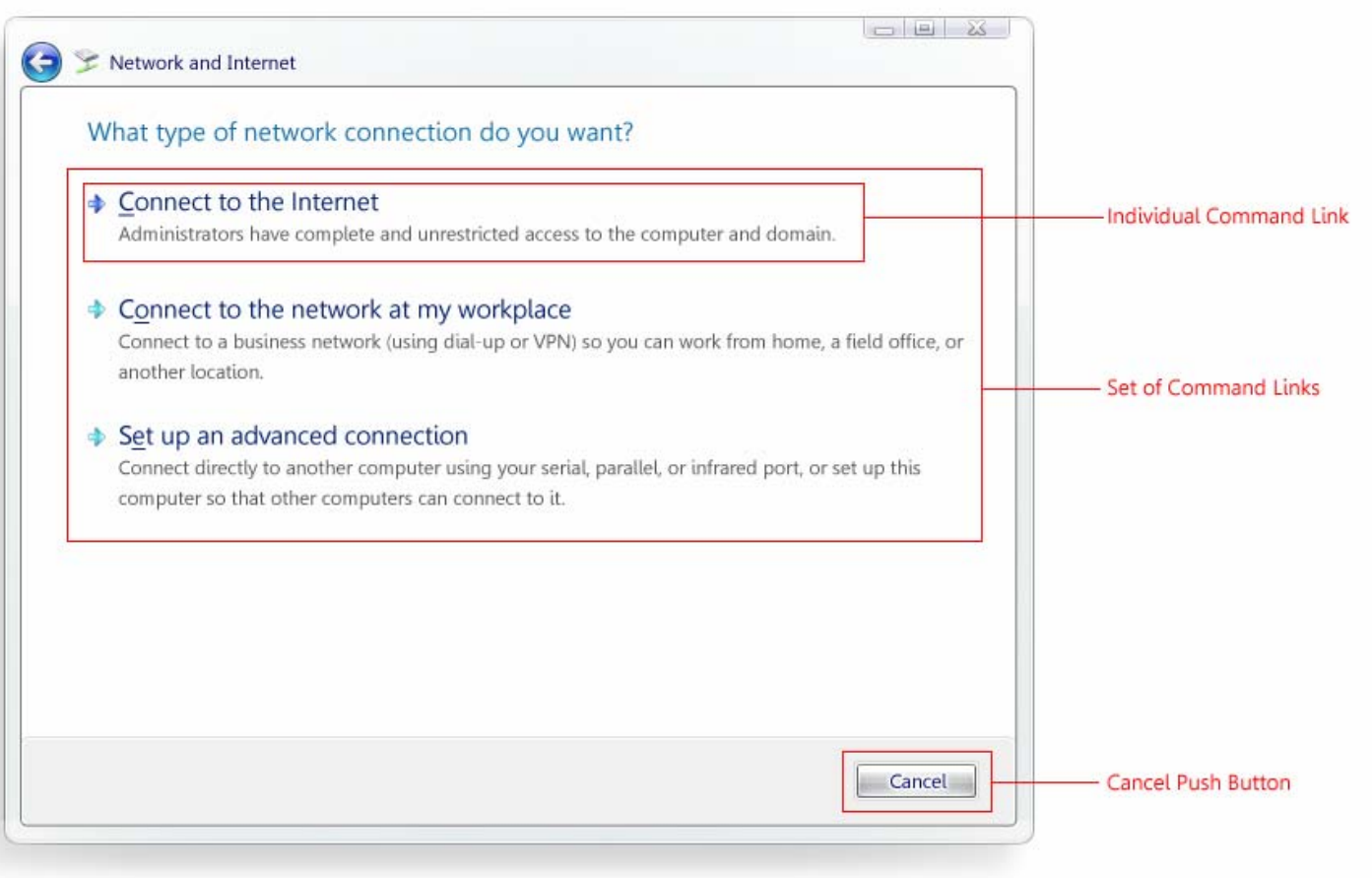


Figure 21 - Task Page with Command Links

Table 6 - Command Links: Summary of Changes

Component	Summary of Changes from Wizard ‘97
Command Links	“Command Links” are used for navigation between pages in a “Command Link” Task Page.

7. Animations & Transitions

7.1 Animations

7.1.1 Overview

7.1.1.1. To give Aero Wizard some extra visual polish, we will add some resizing animations & transitions.

7.1.1.2. All animation will respect SPI_SETUIEFFECTS to disable/enable animations in Aero Wizard.

7.1.2 Resizing Animations

7.1.2.1. For Non-resizable Wizards, we will not show any resizing animations as it is not resizable.

7.1.2.2. For Resizable Wizards, we will introduce a new Window animation when the user progresses to the next Task Page of the Wizard.

7.1.3 Animating to a Larger Page

7.1.3.1. On load, the Wizard will know the size of itself and the (x,y) coordinates of where it was rendered.

7.1.3.1.1. Point A is the upper left (x,y) coordinate.

7.1.3.1.2. Point B is the lower right (x,y) coordinate.

7.1.3.2. Aero Wizard will use Point A as an anchor point for the animation.

7.1.3.2.1. This means Point A never changes for the Wizard unless the user moves the Wizard.

7.1.3.3. When the user progresses to the next page, we calculate the size of the Wizard Frame and find where B' is located (the new B coordinate).

7.1.3.4. The Wizard will then show an animation growing vertically & horizontally to B'.

7.1.3.5. **Figure 22** displays an example of an Aero Wizard growing to a larger page.

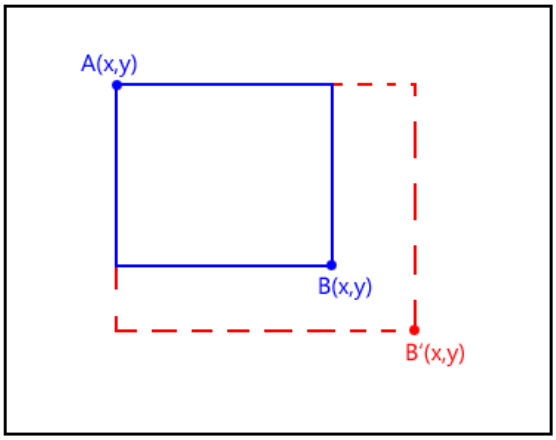


Figure 22 - Wizard animation to a larger page

7.1.4 Animating to a smaller page

7.1.4.1. When Aero Wizard needs to shrink in size to a smaller page, we follow the same logic.

7.1.4.2. Aero Wizard will use Point A as an anchor point for the animation.

7.1.4.2.1. This means Point A never changes for the Wizard unless the user moves the Wizard.

7.1.4.3. When the user progresses to the next page, we calculate the size of the Wizard Frame and find where B' is located (the new B coordinate).

7.1.4.4. The Wizard will then show an animation shrinking vertically & horizontally to B'.

7.1.4.5. **Figure 23** displays an example of an Aero Wizard shrinking to a smaller page.

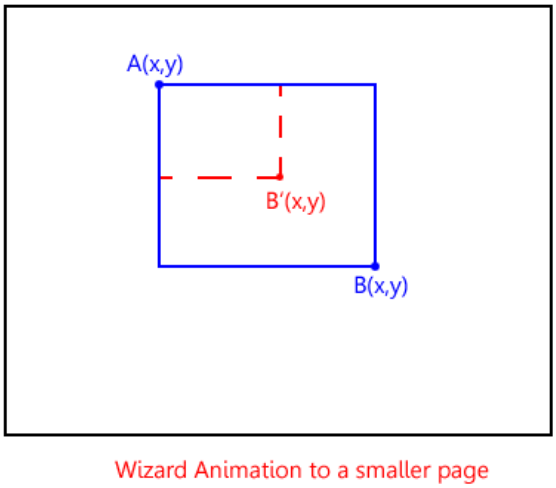


Figure 23 - Wizard animation to a smaller page

7.1.5 Edge Case: Animating when the Wizard overflows the display

7.1.5.1. Since we use the upper left corner as the anchor, there are some cases where growing to a larger page will overflow the wizard on the display. Obviously this is not the desired effect.

7.1.5.2. On load, the Wizard will know the size of itself and the (x,y) coordinates of where it was rendered.

7.1.5.2.1. Point A is the upper left (x,y) coordinate.

7.1.5.2.2. Point B is the lower right (x,y) coordinate.

7.1.5.3. Aero Wizard will use Point A as an anchor point for the animation.

7.1.5.3.1. This means Point A never changes for the Wizard unless the user moves the Wizard.

7.1.5.4. When the user progresses to the next page, we calculate the size of the Wizard Frame and find where B' is located (the new B coordinate).

7.1.5.5. If B' exceeds the available screen real estate, we set B' to the edge of the screen and set Anchor point A to the remaining distance (x or y) we need hence creating A'.

7.1.5.6. The Wizard will then show an animation growing vertically & horizontally to B' and A' at the same time.

7.1.5.7. **Figure 24** displays an example of an Aero Wizard growing to a larger page.

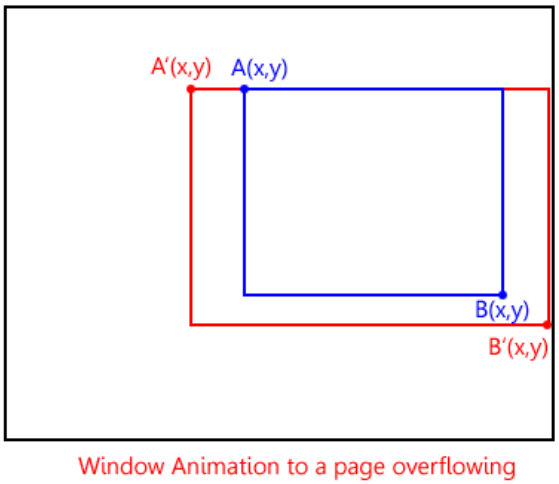


Figure 24 - Wizard animation to a page overflowing

7.1.6 Edge Case: Animating from off screen

7.1.6.1. In some cases the user may push the wizard above, below, to the right, or to the left of the screen making the wizard partially viewable on the desktop.

7.1.6.2. If pushed off the screen, we move anchor point A to the nearest (x,y) coordinate visible on the desktop and calculate B' from the new A anchor position an animate accordingly.

7.1.7 Edge Case: Animating with Multi-mon

7.1.7.1. If the user moves the wizard to span multiple monitors (part of the wizard is on one screen and the rest on another or multiple screens), when the user progresses to the next we keep Anchor A to whatever the user moved it too. Since the Wizard exists in known desktop space we don't need to move the A anchor point.

7.1.8 Maximizing Animation

7.1.8.1. If the Wizard is maximized to full screen we will animate anchor A & B vertically and horizontally to reach the maximized state.

7.1.8.1.1. The system will do this animation. This animation is not specific to Aero Wizard.

7.1.9 RTL

7.1.9.1. For RTL languages, we will flip the animation causing anchor point A to be the upper right corner.

Table 7 – Resizing Animation: Summary of Changes

Component	Summary of Changes from Wizard '97
Wizard Frame	Display a resizing animation for resizable wizards when navigating to new pages.

7.2 Transitions

7.2.1 Overview

7.2.1.1. Aero Wizard will offer simple, smooth transitions of the client area of the wizard when progressing to the next page of a wizard.

7.2.1.2. The client area transition will occur for both resizable & non-resizable Aero Wizards.

7.2.2 Transition Behavior

7.2.2.1. When a user progresses to the next page in a wizard we will show a transition to the new content.

7.2.2.2. The Header Text & Content Area will fade to blank with a fast fade.

7.2.2.2.1. At the same time we will drop any command buttons in the Command Area while keeping the Command Area rendered.

7.2.2.3. The new Header Text & Content Area will fade from blank to the new content.

7.2.2.3.1. At the same time we will show the new command buttons in the Command Area.

7.2.2.4. For resizable wizards, we will fade first then animate, then fade into the new content.

7.2.3 Transition Timing

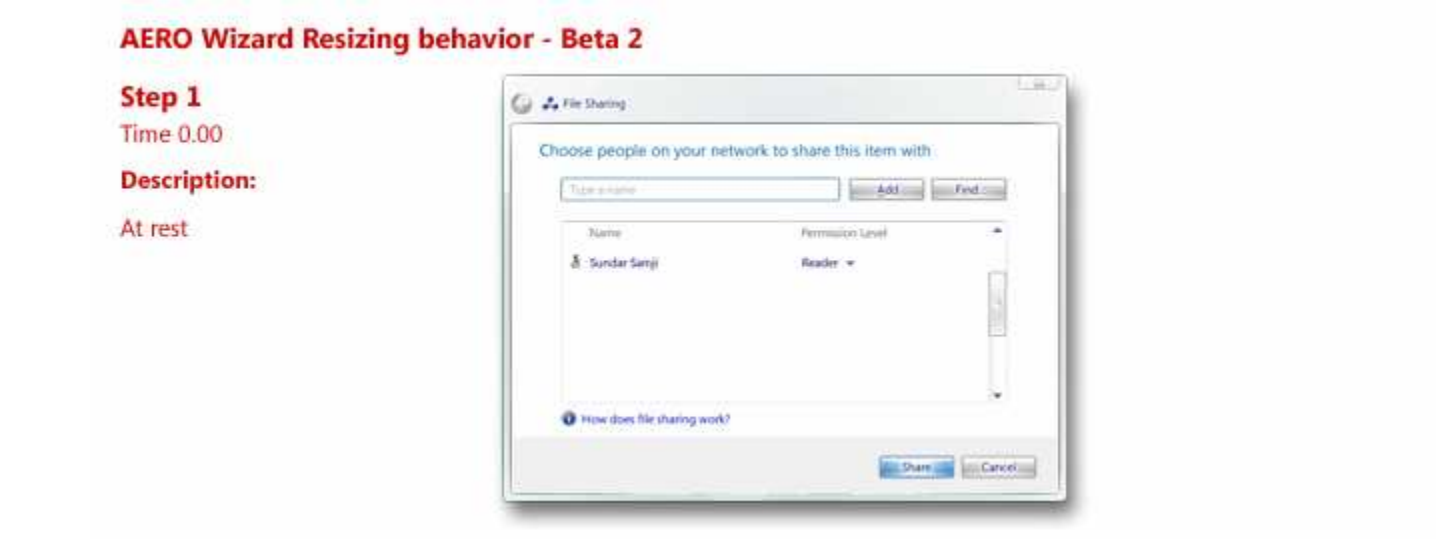
7.2.3.1. The transition timing will specified by the Theme file so it can be used in other places throughout the system

Table 8 – Client Transition: Summary of Changes

Component	Summary of Changes from Wizard '97
Wizard Frame	When progressing to a new page, fade the content to blank & drop the command buttons, animate, then fade the content in and display the new command buttons.

7.3 Animation & Transition Walkthrough

7.3.1 This section gives a visual walk through of how the transition & animation work together in a sample wizard.

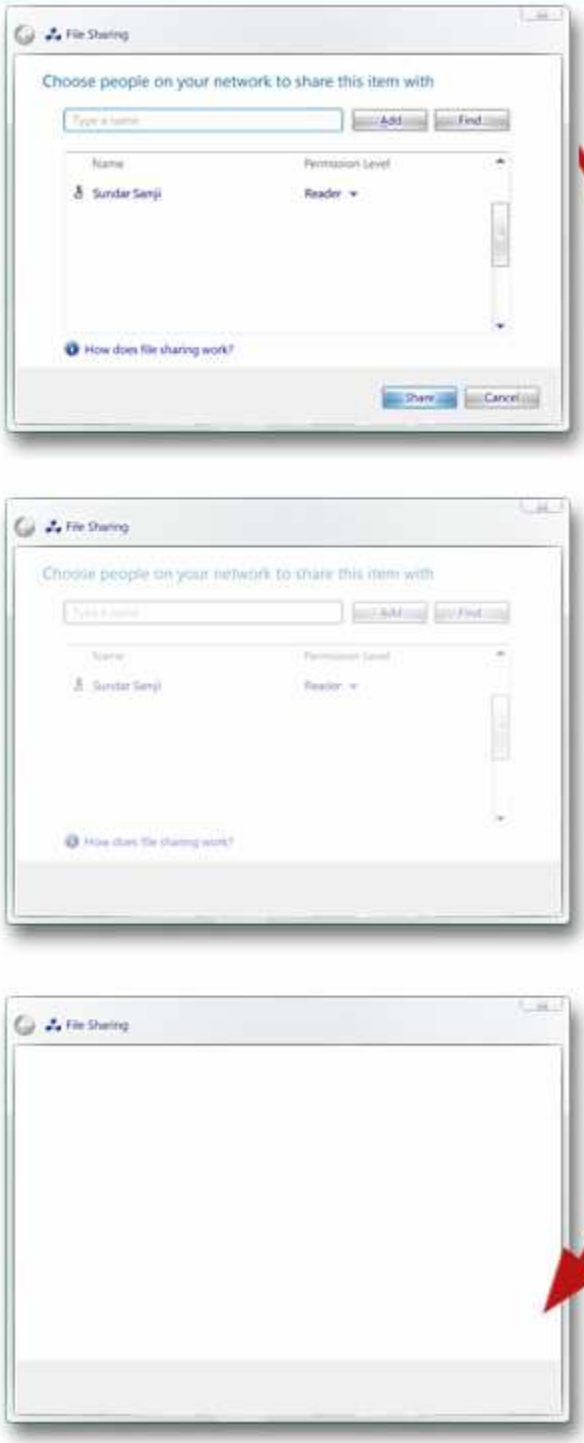


Step 2

Time 0.00 - 0.10 sec

Description:

1) Fade OUT page content to blank
2) Buttons disappear
3) gray command bar area remains



Step 3

Time 0.11 - 0.30 sec

Description:

Animate window to new size

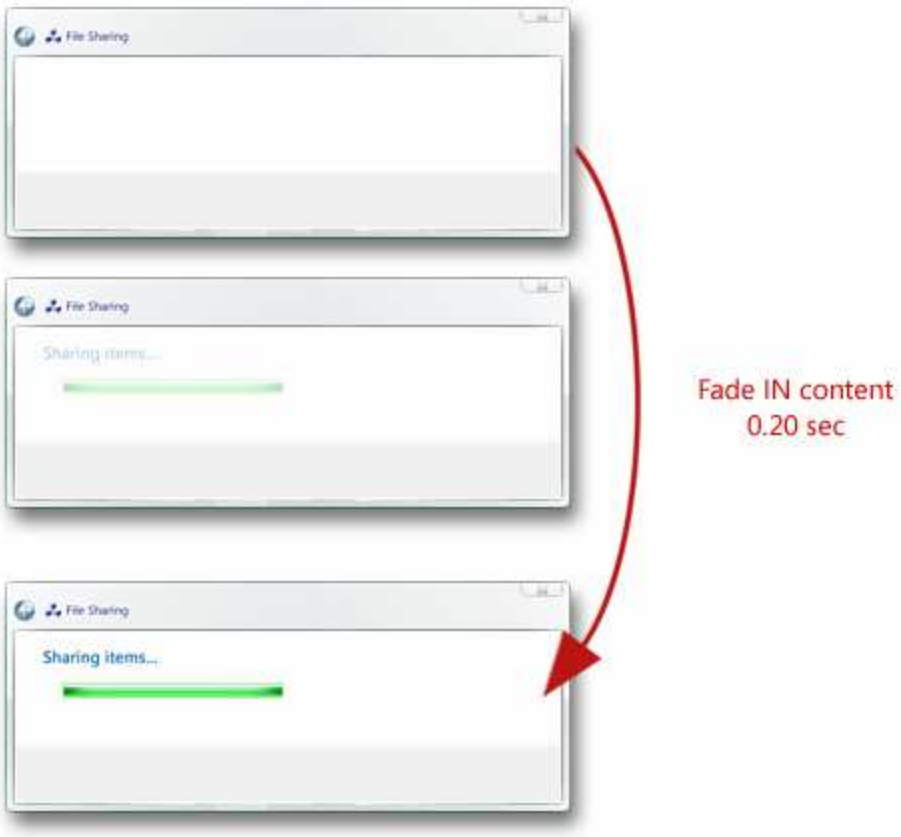
animate window
0.20 sec

Step 4

Time 0.31 - 0.50 sec

Description:

Fade IN new content



Step 5

Time 0.51 - 0.51 sec

Description:

Paint in button(s)

8. Aero Wizard & Themes

8.1 Overview

8.1.1 One of the main goals of Aero Wizard is to update the appearance of the Wizard Framework. The best way to improve the appearance for the LH '06 release as well as future releases is to Theme the Wizard Framework.

8.1.2 By styling the Wizard Framework with Themes, the Theme file can contain colors, sizes and margins which the Framework can use to render the Wizard.

8.1.2.1. This means if we want to change the color of the Wizard Title, we can just change a value in the Theme file and all of the Wizards will have a new Wizard Title color.

8.1.3 The current implementations of the Wizard Framework do not have automatic Theme support, meaning the Framework does not pull any values from the Theme file to render any colors, sizes or margins for the Wizard. Aero Wizard will be the only Wizard Framework that will take advantage of the Theme file.

8.2 Theme Parts & Metrics

8.2.1 This section describes the Theme Parts & Metrics the Wizard Framework will reference.

8.2.2 Definitions

8.2.2.1. Bitmap – an image which could include theme states.

8.2.2.2. Font – contains Font face, style (bold, underlined, etc), size, and color.

8.2.2.3. Color – static color or gradient.

8.2.2.4. Margin – an integer value used for margins.

Table 9 - Theme Parts & Margins for Aero Wizard

Wizard Component	Description	Theme Type	Value Type
Back Button	Wizard Frame Back Button	Theme Part	Bitmap
Title Bar Padding	Spacing between Title Bar Items	Theme Metric	Margin
Wizard Title	Title displayed in the Wizard Frame	Theme Metric	Font
Header Background	Wizard Header Background	Theme Metric	Color
Header Title	Wizard Page title that appears in the header	Theme Metric	Font
Wizard Left Margin	Left Margin of the Wizard	Theme Metric	Margin
Wizard Right Margin	Right Margin of the Wizard.	Theme Metric	Margin
Header Top & Bottom Padding	Margin specifying the padding on the top on bottom of the Header text.	Theme Metric	Margin
Content Area Background	Content Area background	Theme Metric	Color
Content Area Text	Font style, size and color to be used for the Content Area.	Theme Metric	Font
Content Area Bottom Margin	Bottom Padding between the Content & Command Areas	Theme Metric	Margin
Command Area Background	Command Area Background	Theme Metric	Color
Command Button Spacing Margin	Margin for spacing between buttons	Theme Metric	Margin
Command Button Padding	Top and Bottom margin for the command buttons	Theme Metric	Margin

8.3 Colors & Styles

8.3.1 This section covers all of the colors & styles for Aero Wizard components

Table 10 - Colors & Styles

Component	Style
Title bar	Font: Segoe UI
Header Title (Main Instruction)	Font: Segoe UI Size: 12pt Color: RGB (19,112,171) Hex #1370AB
Header Background	RGB (255, 255, 255) Hex #FFFFFF
Content Area Text	Font: Segoe UI Size: 9pt Color: RGB (87,87,87) Hex # 575757
Content Area Background	RGB (255, 255, 255) Hex #FFFFFF
Command Area Background	RGB (240, 240, 240) Hex #F0F0F0
Button Text	Font: Segoe UI Size: 9pt (96 dpi) Color: RGB (0,2,98) Hex # 000262

9. Aero Wizard & Classic Visual Style

This section covers requirements for Aero Wizard when running under Classic Visual Style

9.1 Visuals

9.1.1 General

9.1.1.1. Aero Wizard will render with system metric colors.

9.1.2 Caption Area

9.1.2.1. The caption area will render like a normal Window, displaying the Wizard Icon & Title.

9.1.3 Title Bar

9.1.3.1. Aero Wizard will continue to show the Wizard Icon & Title in the Title Bar as well.

9.1.3.2. Aero Wizard will use a “Classic” back button bitmap when rendered in Classic Visual Style.

9.1.3.3. Title Bar background will render based on system metric colors.

9.1.4 Header

9.1.4.1. The Header title will render using the system font with a bold style.

9.1.4.2. Header background will render based on system metric colors.

9.1.4.3. The top & bottom margins will continue to be used.

9.1.5 Content Area

9.1.5.1. Content Area text will render with the standard system font.

9.1.5.2. Content Area background will render with standard system metric colors.

9.1.5.3. Developers must specify fallback colors for the task pages if they are using DUI.

9.1.6 Command Area

9.1.6.1. Command Area will render Classic Command Buttons.

9.1.6.2. Command Area background will render with standard system metric colors.

9.2 Classic Visual Style Spec

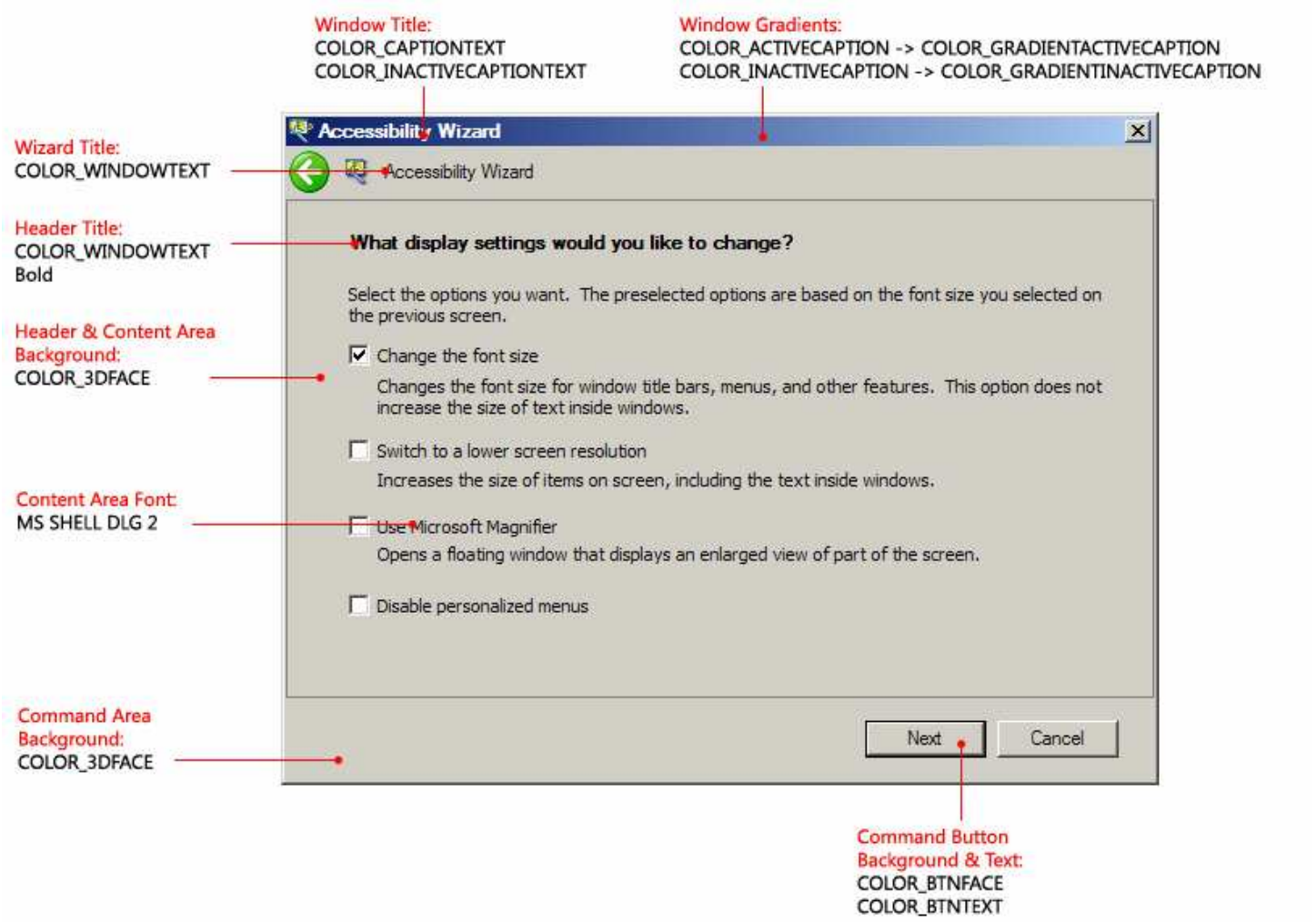


Figure 25 - Classic Visual Style Spec

10. Basics

This section covers requirements for the Longhorn Basics.

10.1 Accessibility

10.1.1 General

10.1.1.1. Aero Wizard shall support High DPI - All Text, Icons, and Controls will scale appropriately with the set DPI.

10.1.1.1.1. High DPI Theme parts will be available for the Back button.

10.1.1.1.2. All text will scale based on the DPI.

10.1.1.1.3. All controls will scale based on the DPI.

10.1.1.2. Aero Wizard shall support High Contrast Mode.

10.1.1.2.1. Since Aero Wizard works in Classic Visual Style, High Contrast Mode will work automatically (as implemented in XP).

10.1.1.3. All Aero Wizard elements will have MSAA support, thus supporting screen readers.

10.1.2 Keyboard Navigation

10.1.2.1. Aero Wizard supports normal keyboard navigation (Tab, Arrow Keys).

10.1.2.2. Keyboard navigation for the Back Button

10.1.2.2.1. Alt + Left

10.1.2.2.2. Tab to the Back Button and press “ENTER”

10.1.2.3. Users can close the Wizard by pressing ESC or ALT+F4 (standard close). This acts the same as pressing the “x” caption button or the “Cancel” Command Button.

10.1.2.4. Aero Wizards will follow the same Tab Order.

10.1.3 Tab Order

10.1.3.1. The Tab order for Aero Wizard is as follows:

10.1.3.1.1. Content Area

10.1.3.1.2. Next

10.1.3.1.3. Finish

10.1.3.1.4. Cancel

10.1.3.1.5. Back

10.1.3.2. If any of the buttons are missing on a Task Page, the Tab order will remain the same but will skip over missing elements.

10.1.4 MSAA Properties

10.1.4.1. The following MSAA/DUI properties need to be set for each Aero Wizard component.

10.1.4.1.1. Shortcut – Set the Keyboard Shortcut

10.1.4.1.2. Role – Should be set according to MSAA role constant. For example an OK button should be set to PushButton. Refer to IAccessible::get_accRole().

10.1.4.1.3. State – Should be set according to MSAA state constant flags. For example, a checkbox we should know whether it is Checked or not. Refer to IAccessible::get_accState().

10.1.4.1.4. Name – For buttons, value for custom buttons should be the text on the control itself. For example, the OK button should be named OK. So when a button is localized the name will be localized too. Refer to IAccessible::get_accName().

10.1.4.1.5. Description – A text description of a control. Refer to IAccessible::get_accDescription().

10.1.4.1.6. Value – An optional value of a control. Refer to IAccessible::get_accValue().

10.1.4.1.7. Default Action – Default action of a control. For example, a button default action should be Press. Refer to IAccessible::get_accDefaultAction().

10.1.4.2. Only Role, State, and Name are required for automation. Other properties are normally used by accessibility tools such as Narrator.

10.1.4.3. **Table 11** describes all MSAA Properties for Aero Wizard.

Table 11 - MSAA Properties for Aero Wizard

Component	MSAA Properties
Back Button	**Shortcut:** ALT + B **Role:** ROLE_SYSTEM_PUSHBUTTON **State:** STATE_SYSTEM_NORMAL **Name:** "Back" (or localized equivalent) **Description:** "Returns to the previous page" **Value:** N/A **Default Action:** Press
Wizard Icon	**Shortcut:** N/A **Role:** ROLE_SYSTEM_GRAPHIC **State:** STATE_SYSTEM_NORMAL **Name:** "Wizard Icon" (or localized equivalent) **Description:** "Wizard Icon" (or localized equivalent) **Value:** N/A **Default Action:** N/A
Wizard Title	**Shortcut:** N/A **Role:** ROLE_SYSTEM_TITLEBAR **State:** STATE_SYSTEM_TEXT **Name:** N/A **Description:** Displays the name of the wizard and contains controls to manipulate it **Value:** Title of the Wizard (or localized equivalent) **Default Action:** N/A
Header Title	**Shortcut:** N/A **Role:** ROLE_SYSTEM_STATICTEXT **State:** STATE_SYSTEM_TEXT **Name:** Header Title (or localized equivalent) **Description:** Display the name of the page **Value:** N/A **Default Action:** N/A
Next Button	**Shortcut:** ALT + N **Role:** ROLE_SYSTEM_PUSHBUTTON **State:** STATE_SYSTEM_PRESSED **Name:** "Next" (or localized equivalent) **Description:** N/A **Value:** N/A **Default Action:** Press
Custom "Next" Button	**Shortcut:** ALT + Whatever the Mnemonic is (usually the first letter) **Role:** ROLE_SYSTEM_PUSHBUTTON **State:** STATE_SYSTEM_PRESSED **Name:** Label text (or localized equivalent) **Description:** N/A **Value:** N/A **Default Action:** Press
Cancel Button	**Shortcut:** N/A **Role:** ROLE_SYSTEM_PUSHBUTTON **State:** STATE_SYSTEM_PRESSED **Name:** Cancel (or localized equivalent) **Description:** N/A **Value:** N/A

	Default Action: Press
Finish Button	**Shortcut:** ALT + F **Role:** ROLE_SYSTEM_PUSHBUTTON **State:** STATE_SYSTEM_PRESSED **Name:** “Finish” (or localized equivalent) **Description:** N/A **Value:** N/A **Default Action:** Press
Custom “Finish” Button	**Shortcut:** ALT + Whatever the Mnemonic is (usually the first letter) **Role:** ROLE_SYSTEM_PUSHBUTTON **State:** STATE_SYSTEM_PRESSED **Name:** Label Text (or localized equivalent) **Description:** N/A **Value:** N/A **Default Action:** Press
Dialog Window	**Role:** ROLE_SYSTEM_WINDOW **State:** STATE_SYSTEM_RESIZABLE, STATE_SYSTEM_MOVABLE, STATE_SYSTEM_FOCUSABLE **Name:** Window Title text (or localized equivalent) **Description:** N/A **Value:** N/A **Default Action:** N/A

11. Aero Wizard Summary of Changes

Component	Summary of Changes from Wizard '97
	Wizards will show up in the Task Bar.
	Aero Wizard offers a flag for resizing the Wizard.
	Add support to resize the Frame to the size of the Property Page.
	Add support to keep the placement of the wizard when the user advances through the wizard.
	Do not resize the Wizard when the Wizard is Maximized.
	Resize -> Next -> Back – Save the state of the resized page.
	Command Buttons are pegged to the lower right corner on resize.

Component	Summary of Changes from Wizard '97
Aero Wizard Title Bar	"Back" button of the Wizard Framework is moved to the top of the Wizard.
	"Wizard Icon" is displayed in the bar next to the back button.
	The style of the "Wizard Title" is specified by the Theme File.
	"Wizard Title" appears in the client area title bar, not the non-client area.

Component	Summary of Changes from Wizard '97
Aero Wizard Header	Header background color should be pulled from the Theme file.
	The Top of the Header should be rendered with rounded corners when themes are enabled. This should be pulled from the Theme file.
	"Header Title" is themed. The style & size is specified by the Theme File.
	"Header Sub-Title" is no longer displayed.
	A Header Background bitmap can be defined as a nine-grid.
	Developer can specify a custom color for the Header text when a Header Bitmap is defined.
	A top and bottom margin will be implemented for the header.
	"Header Title" can wrap over multiple lines.

Component	Summary of Changes from Wizard '97
Aero Wizard Content Area	"Content Area" background will respond to a Theme change. For Aero, the background will be white.
	"Content Area" controls & text will respond to Themes. If static text is present, it will get the current font style & size from the current Theme File.
	Developers can use any size content area they wish.
	Developers can specific "No Margin" for the content area.

Component	Summary of Changes from Wizard '97
Command Area	The Command Area should render rounded corners on all sides of the Command Area. This should only appear when Themes are enabled.
	"Next" & "Finish" Command Buttons can be renamed.
	"Next", "Finish", "Cancel", and Back buttons can be hidden.
	If all buttons are hidden, the command area is hidden.
	Button size normalization.

Component	Summary of Changes from Wizard '97
Command Links	Flag specifying a "Command Link" type Task Page. No Command Area will be displayed.
	"Command Links" are used for navigation between pages in a "Command Link" Task Page.

Component	Summary of Changes from Wizard '97
Wizard Frame	Display a resizing animation for resizable wizards when navigating to new pages.

Component	Summary of Changes from Wizard '97
Wizard Frame	When progressing to a new page, fade the content to blank, animate, then fade the content in.

12. Appendix A - Visual Specifications

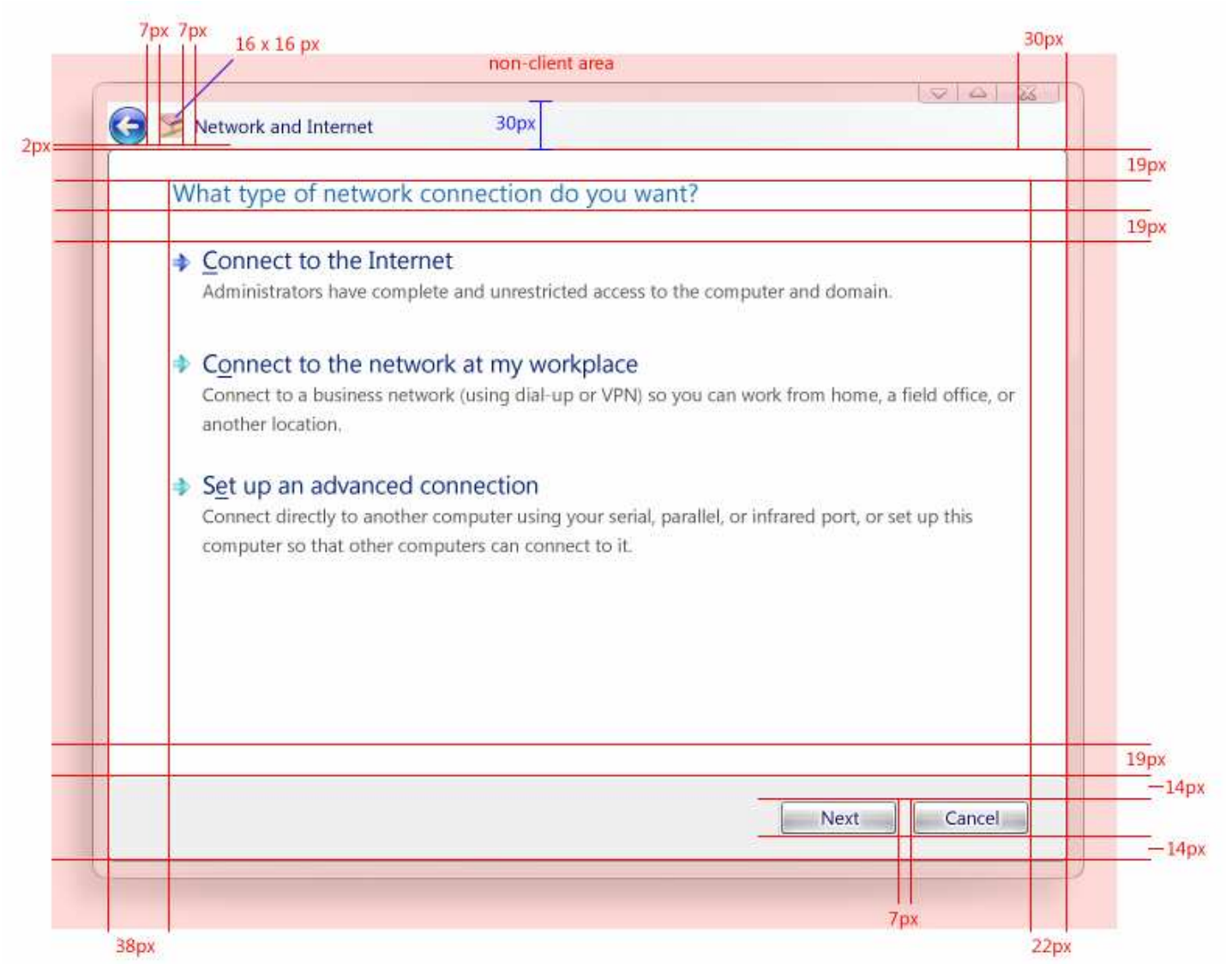


Figure 26 - Aero Wizard Visual Spec

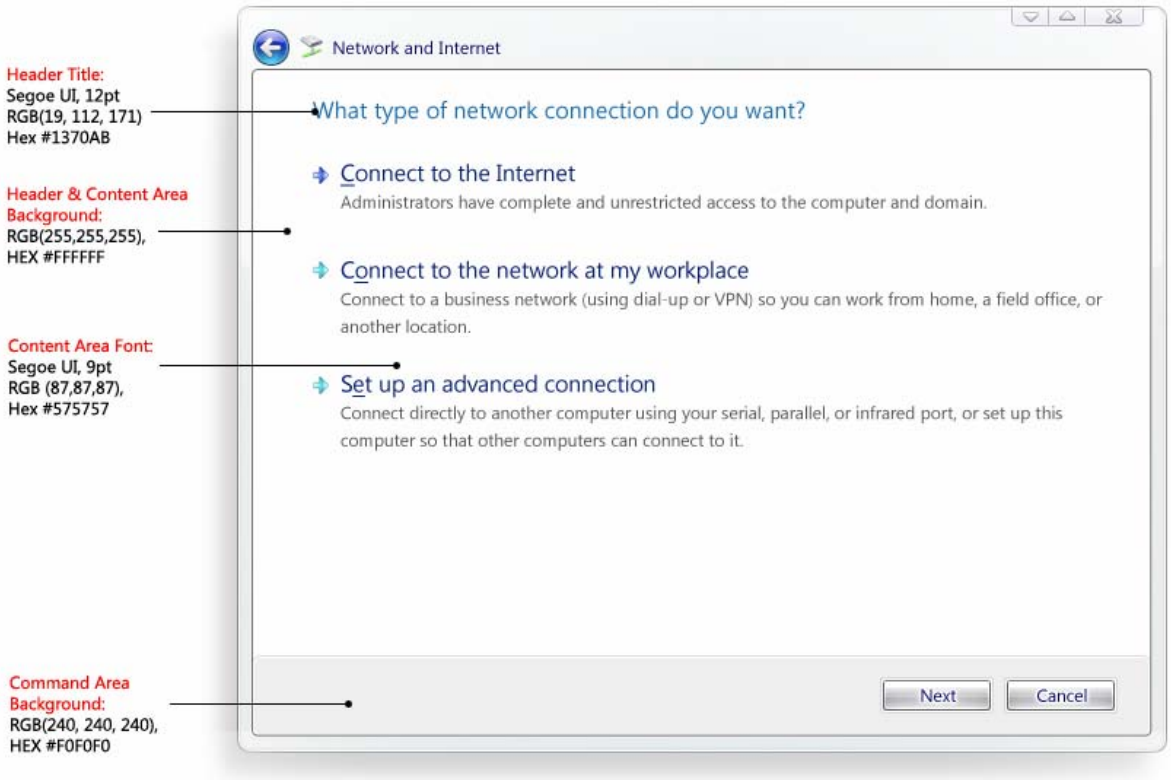


Figure 27 - Aero Wizard Fonts & Colors

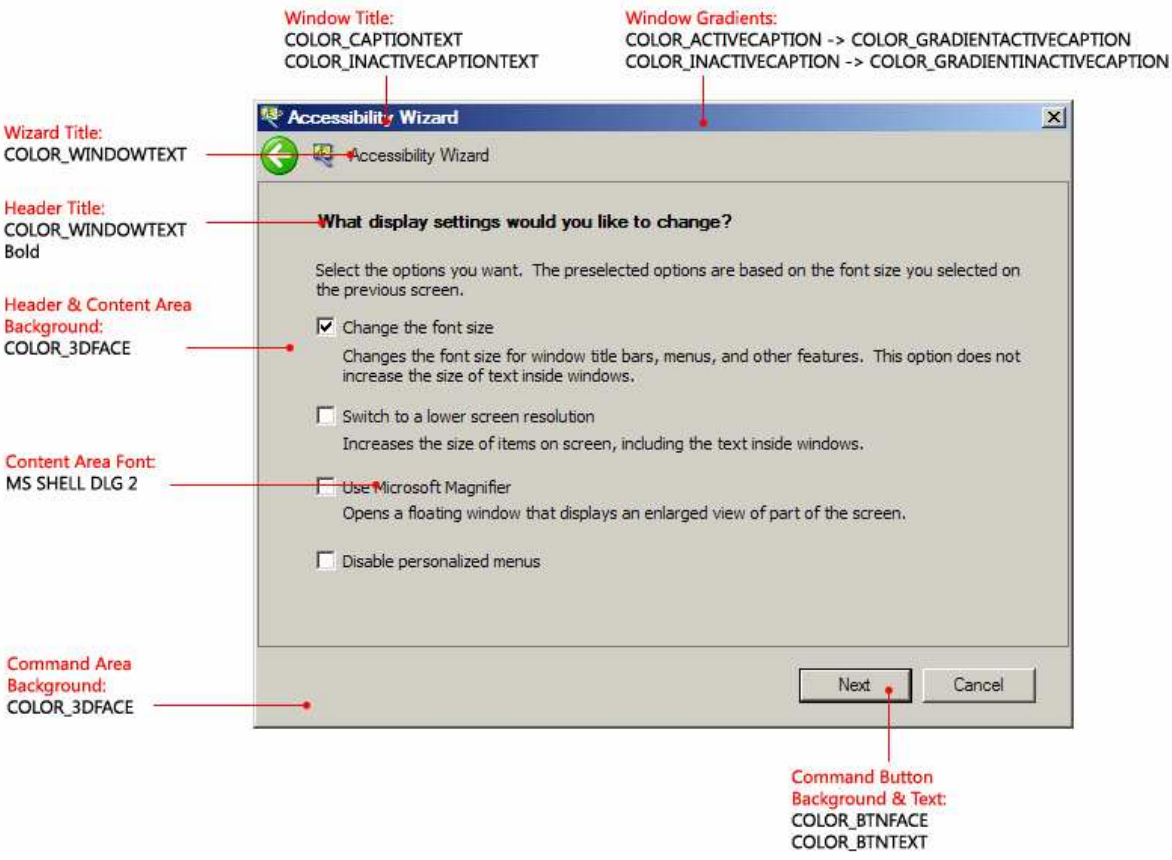


Figure 28 - Aero Wizard in Classic Visual Style

13. Appendix B – Decisions/Q&A

13.1 Aero Wizard "Back" Button

Q: Why did you move the Wizard "Back" button to the upper left corner?

Traditionally Wizards always displayed all buttons in the bottom right of the Wizard. We decided to move the Back button to the upper right corner for the following reasons:

1. Consistency with Explorer
    a. Explorer has a "navigation area" where the back and forward buttons are in the upper left corner. The new navigation buttons are larger and easily identifiable. If users understand this, they can understand that the back button is in the upper left in a wizard as well.
2. Users should not have to go back in a wizard often
    a. The point of a wizard is to walk through users through a set of tasks in a certain order. Users shouldn't have to navigate back very frequently when running a wizard. Because of this, it doesn't need to be in a location a user can access as quickly as the next button.
3. Consistency with IUI navigation
    a. All IUI based control panels use the navigation buttons in explorer. Users will build a natural affordance which can be used in Wizards as well.

Q: How did it test with users?

The usability team tested this new behavior a lot when it was first implemented. Many users asked "why did you move it" but all were able to find it and use it. 85% were able to find the back button immediately while the other 15% needed to look for it.

13.2 DCRs/Cuts

13.2.1 Auto resizing

13.2.1.1. Auto resizing was removed from Aero Wizard Framework because it tested very poorly with users. Many users found this behavior distracting as they needed to move their mouse to different locations for the "Next Button" for each page.

13.2.2 Animations & Transitions

13.2.2.1. We felt the animations & transitions weren't high enough priority for this particular feature.