General Guidelines

Microsoft Windows has built-in accessibility capabilities that are available to all users. The standard controls (e.g., menus, buttons, lists, windows) built into the Windows operating system already support accessibility features (i.e., the standard interface objects in the Windows environment have accessibility enabled by default). Therefore, a software application should be designed to not override this built-in behavior. Instances where software should provide explicit accessibility support are the following: providing a custom-designed interface specific to the application domain of the software, and creating custom controls or enhancing the normal behavior of standard controls.

Providing a custom designed interface specific to the application domain of the software

Windows provides accessibility features within individual controls. The software developer must consider the interoperability of controls between the software, assistive technology, and the operating system environment. However, the accessibility issues related to how all the controls are combined into a software interface are the responsibility of the designer and developer. For example, defining the menu; menu item shortcut keys; and window layout of controls for the software.

Creating custom controls or enhancing the normal behavior of standard controls

There are times when the standard controls provided by Windows cannot mimic the preferred interface design (e.g., an interface to duplicate the look and feel of an instrumentation panel). In this case, custom controls may be purchased or created from scratch. Care must be taken to ensure that all controls or classes purchased or built from scratch follow the Section 508 guidelines for accessibility compliance.

The problem with creating custom controls is that assistive technology may have difficulty identifying them (i.e., assistive technology require specific information in order to work successfully with screen elements). The methods that most assistive technology uses to identify screen elements are as follows:

- Microsoft Active Accessibility: Assistive technology uses the Active Accessibility interface to retrieve information and get status change notifications directly from objects that support it. All standard Windows controls use Active Accessibility.
- Window Messaging: If an object has a window handle, an aid can be informed via a message when objects are created or destroyed. The AT can also look at the window class and retrieve standard information. All standard Windows controls utilize messaging.

- Off-Screen Model: Assistive technology monitors drawing operations (e.g., text and graphics) on the screen and builds a model of the screen's contents. This method relies on complex heuristics. It does not provide meaning to the output, nor is it dependable. It should be used as a last resort.

Therefore, custom controls should utilize the Microsoft Active Accessibility interface and messaging capabilities within Windows. The information that objects should provide are as follows:

- Name, location, type, and associated value(s).
- Parent control (i.e., when controls are contained within other controls).
- Logical order in which controls are navigated.
- Event notification (e.g., when controls are created, destroyed, gain/lose focus, status changed (e.g., checked/unchecked), activated (e.g., button pressed)).

When developing custom controls, the developer must work at the operating system level and be aware of how the operating system supports the accessibility features within the standard controls so they can be mimicked in the custom controls. This requires intimate knowledge of the Windows Application Programming Interface (API). The most effective way for custom controls to be accessibility enabled is as follows:

- Create COM objects representing individual custom controls or groups of controls that properly support the "IAccessible" interface.
- Call "NotifyWinEvent" when controls are created or destroyed, gain or lose focus, or otherwise change state.
- Handle the appropriate messages (e.g., WM_GETOBJECT, WM_GETTEXT) used to query properties of the object or objects.

Other general guidelines

- Use standard Windows controls whenever possible because they are compatible with assistive technology.
- Maintain compatibility with assistive technology when designing the overall interface and custom controls.

- Incorporate the requirement for developing accessible software into your overall software lifecycle (e.g., planning, analysis, design, coding, testing).
- Provide a customizable interface that accommodates a wide variety of user needs and preferences (e.g., colors, fonts).
- Provide flexibility in using a variety of input methods (e.g., keyboard, mouse) and output methods (e.g., color, sound, images and text).
- Maintain consistency with the system-wide settings specified within the operating system (e.g., color schemes, contrast, font sizes, keyboard/mouse repeat rates and sensitivity).
- Identify accessibility issues inherent to the development environment, and purchase or develop workarounds.
- When phasing in accessibility compliance, prioritize the enhancements by identifying the features that affect the largest number of users, that are used most frequently, and that are required rather than optional.
- Instead of just offering preferences that need to be set by the user, query the assistive technology being used and configure the software automatically. For example, check Windows system information (i.e., SystemParameterInfo) to determine when a screen reader is in use (i.e., SPI_GETSCREENREADER). Also, Windows will invoke a message when system information is changed so that the software can be reconfigured.
- Maintain action consistency (i.e., similar actions have a similar interface).
- Make actions reversible whenever possible; otherwise, request confirmation.
- Use common dialog boxes whenever possible; otherwise, follow common dialog box conventions when developing custom dialogs.
- Follow guidelines for ordering commands on menus.
- Use standard toolbar label images and text when supporting common actions.
- Avoid drawing over a variegated background, unless there is an option to omit the background. Although this technique can be pleasing to some, it can be very difficult to read for others. Also, avoid jagged edges (e.g. italicized fonts).
- Provide counts when displaying items in a list.

- Avoid triggering actions on user navigation in the interface.
- Test for interoperability with assistive technology.

§1194.22(a)
A text equivalent for every non-text element shall be provided (e.g., via "ALT", "Longdesc", or in the element content).

`<IMG SRC="someimage.gif" ALT="This is a description of some image.">` shows coding of an image that has alternate text associated with it

Do images exist on the web page?

If yes, check for some variation of:

EX: `<img src="someimage.gif" alt="This is a description of the image">`

Do spacer images exist on the web page? (Look for images that are used to move or space over other images, text, etc.)

If yes, check to see if each image that is used as a spacer has a blank alt tag. Having `alt=""` allows assistive technologies to skip over the image.

Ex: `<img src="shim.gif" alt="">`

Do button images exist on the web page? (Look for images that are used as buttons in forms, navigation, etc.)

If yes, check to see if each image that is used as a button has alternate text associated with it.

Ex: `<input type="image" src="submit.gif" alt="Submit Button">` or `<a href="nextpage.html"><img src="next.gif" alt="Click here to go to the next page"></a>`

§1194.22(b)
Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.

§1194.22(c)
Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.

§1194.22(d)
Documents shall be organized so they are readable without requiring an associated style sheet.

Are style sheets being used?

Ex:

```
<style type="text/css">
body {
   font-family: Verdana;
   font-size: 12px;
   color: #000;
      }
</style>
```

or

```
<link rel="stylesheet" href="default.css" type="text/css">
```

If yes, turn off style sheets to determine if the information can still be read.

To turn off style sheets:

- In Netscape, click on *EDIT, PREFERENCES, ADVANCED,* make sure *ENABLE STYLE SHEETS* is not checked, then click *OK*.
- In Internet Explorer, click on *TOOLS, INTERNET OPTIONS, ACCESSIBILITY,* check *IGNORE FONT STYLES AND IGNORE FONT SIZES,* then click *OK, OK.*
- Go back to the browser window and reload the web page. Can the information be seen and read?

§1194.22(e)
Redundant text links shall be provided for each active region of a server-side image map.

View the source code of the web page.

Are server-side image maps being used? (Look for `<IMG src="server.gif" alt="Server-side image map" ismap>`.)

If yes**,** check to see if the image map is accompanied by a set of redundant text links (i.e., a separate page of links or a list of links somewhere else on the page).

§1194.22(f)
Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

§1194.22(g)
Row and column headers shall be identified for data tables.

Are data tables being used? (Look for the `<table>` HTML tag.)

If yes**,** determine if row and column headers are being used. (EX: `<tr>`, `<th>` and `<td>`).

§1194.22(h)
Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.

Are tables being used? (Look for the `<table>` HTML tag.)

If yes, determine if data cells and header cells are associated:

Scope Example:

```
<table>
<tr>
      <td> </td>
      <th scope="col">Name</th>
      <th scope="col">Phone Number</th>
      <th scope="col">Address</th>
</tr>
<tr>
<th scope=”row”>1</th>
      <td>John Doe</td> <td>(123) 456-7890</td> <td>123
      Main St.</td>
</tr>
<tr>
      <th scope=”row”>2</th>
      <td>Jane Doe</td> <td>(123) 456-7890</td> <td>123
      Main St.</td>
</tr>
<tr>
      <th scope=”row”>3</th>
      <td>John Doe Jr.</td> <td>(123) 456-7890</td>
      <td>123 Main St.</td>
</tr>
</table>
```

ID and Header Example:

```
<table>
<tr>
      <th id=”name”>Name</th>
      <th id="phone">Phone</th>
      <th id="address">Address</th>
</tr>
```

```
<tr>
      <td headers="name">John Doe</td>
      <td headers="phone">(123) 456-7890</td>
      <td headers="address">123 Main St.</td>
</tr>
<tr>
      <td headers="name">Jane Doe</td>
      <td headers="phone">(123) 456-7890</td>
      <td headers="address">123 Main St.</td>
</tr>
<tr>
      <td headers="name">John Doe Jr.</td>
      <td headers="phone">(123) 456-7890</td>
      <td headers="address">123 Main St.</td>
</tr>
</table>
```

§1194.22(i)
Frames shall be titled with text that facilitates frame identification and navigation.

Are frames being used?

If yes, determine if the frame is titled descriptively.
(EX: `<FRAME src=”MainNavigation.html” title=”Main Navigation Bar”>`)

§1194.22(j)
Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.

§1194.22(k)
A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.

§1194.22(l)
When web pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.

If developers do not put functional text with scripting languages, the screen reader might read some of the script to the user, which can come in a combination of letters and numbers, and won’t be comprehensible to the user. For example, `<a href="javascript:someFunction();">Perform Some Function</a>`, will read “perform some function” to the user. If the words

“Perform Some Function” did not exist, then the user would not know what the link was going to do.

§1194.22(m)
When a web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with 1194.21 (a) through (l).

The applet needs to comply with the regulations in Section 508 sub-part 1194.21. The applet should be tested as a software application to ensure that it complies with all the required regulations of Section 508 – 1194.21.

§1194.22(n)
When electronic forms are designed to be completed on line, the form shall allow people using assistive technology to access the information, field elements and functionality required for completion and submission of the form, including all directions and cues.

Do any electronic forms exist on the web page?

If yes**,** determine if all form controls are explicitly associated with labels.

```
<tr>
      <td><label for="name">Name:</label></td>
      <td><input type="text" name="name" id="name"></td>
</tr>
```

§1194.22(o)
A method shall be provided that permits users to skip repetitive navigation links.

§1194.22(p)
When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.