Design Guidance

Accessibility Checklist

Thursday, 5 July 2007
Version 1.0.0.0

Prepared by
Microsoft

Microsoft®

TABLE OF CONTENTS

1 INTRODUCTION

Accessibility is an essential requirement for the success of the clinical applications. This requirement is underlined by commitment to equality, an obligation under the anti-discrimination legislation.

This document comprises an extension to the requirements detailed in the following design guidance documentation:

- *Design Guidance – Accessibility Principles* **{R1}**

This new document expands each of the original 10 requirements defined in the *Accessibility Principles* document into a number of specific checkpoints, and details a testing and verification methodology for each of the new checkpoints.

1.1 Purpose of the Document

The information contained in this document provides the following:

- An increase in the specificity and clarity of the accessibility requirements to provide designers and developers with an increased level of detail to design and code against
- A suggested testing and verification methodology to support checking against each of the checkpoints, facilitating compliance and verification of achieved level of compliance
- A description of the relationship of the accessibility checkpoints to existing accessibility standards to make it easier to understand the genesis of the accessibility requirements and how these compare to other standards

1.2 Working Assumptions

In developing this document the following assumptions were made:

- The 10 requirements and their associated checkpoints will make no inherent assumption about specific technical implementation of user interface components. However, in considering requirements for web-based implementation, it will be assumed that the most up to date version of browser software is available and JavaScript is available
- The testing methodologies assume a Windows XP platform

1.3 Scope

1.3.1 Conformance to other Accessibility Standards

Conformance to the accessibility requirements and checkpoints detailed in this document will not ensure that all aspects of regulatory requirements from other standards are adhered to. Separate validation against the specific standard must be made before making conformance claims to that standard. For example, a web application should be specifically validated against the *Web Content Accessibility Guidelines version 1.0*[1] (WCAG 1.0) **{R2}** before making claims against the various conformance levels that it defines.

[1] At the time of writing, the Web Content Accessibility Guidelines version 1.0 (WCAG 1.0) is the current recommendation from the World Wide Web Consortium (W3C). It dates from 1999 and is due to be replaced by WCAG 2.0 in the near future: http://www.w3.org/TR/WAI-WEBCONTENT

Web applications that conform to the required Web Content Accessibility Standards (currently all priority 1 and priority 2 checkpoints of WCAG 1.0) do not need to be separately validated against the checkpoints defined in this document. WCAG 1.0 is more specific to web implementations than the checkpoints contained in this document and therefore takes precedence.

1.3.2 Limits to the Current Specificity of the Requirements and Checkpoints

The checkpoints as defined in this document represent a generalised framework derived from other publicly available sources (see section 2). It is expected that the checkpoints will be refined to more specifically match the requirements for the clinical application user interface in the future. The refinement will recognise that the deployment of clinical applications will be under more controlled conditions than the Web and general publicly-available software. This permits the formulation of a more constrained and practical definition of context that recognises knowledge of the following factors:

- **People** – The range of abilities and disabilities of healthcare workers and how these relate to the different job roles and, therefore, interactions with specific system features or components
- **Environment** – The range of physical locations, and situational and environmental circumstances where machines running the applications will be deployed and how this relates to interactions of specific system features or components
- **Equipment** – The range of specified hardware and software devices that will be supported and how this relates to interactions with specific system features or components

Information on all of the above factors needs to be gathered and analysed. Note that there are also interactions between people, environment and equipment that specify the full context of use. These complex interactions will need to be fully researched. The information will be used to set accessibility requirements within the context of how they will actually be used in practice.

This is important because:

For particular specified components of healthcare applications, certain accessibility requirements will not apply, and will therefore be excluded from specific aspects of any validation or conformance process.

As an example, it might be confirmed through continuing research that healthcare workers who are blind are never required to rely on information that is only available in an X-ray image. Given this context, no *equivalent* for this non-text element (the X-ray) would be required. A clear indication that it was an X-ray image would be required, but a description of what the X-ray showed (the equivalent) would not be required.

In relation to the checkpoints defined in the current version of this document, this means that failure to meet the conditions of one or more checkpoints will not necessarily imply failure to meet accessibility requirements. However, each checkpoint must be considered and justification for each failure should be supplied (see section 1.3.4).

1.3.3 Suggested Testing Methodology

The testing methods detailed in this document suggest one way of performing accessibility checks and assume a Windows XP platform. This assumption recognises that the majority of clinical applications are Windows based and the majority of adaptive technologies in use are also Windows based. The accessibility requirements and checkpoints themselves are not technology dependent, and the basic testing approach suggested in this document is transferable to other non-Windows platforms.

Given your development environment and available tools, more appropriate ways of testing may be used as long as they achieve the same result as indicated for each checkpoint. When the range of hardware and software (including adaptive technologies, such as screen readers) that will be supported is defined, future versions of this document may define a rigid testing methodology.

Specific instructions on testing tools (for example, screen readers) are beyond the scope of this version of the document. Step by step instructions may appear in future versions; please refer to specific tool documentation for instructions on how to use each of the tools (see Appendix A).

Although beyond the scope for this document it is recommended that all applications are subject to user testing and that the testing should include people with disabilities who use a variety of adaptive technologies.

1.3.4 Conformance of the Checkpoints

The following limits to conformance expectations currently apply:

- Detailed accessibility requirements may be set out by the functional specification for a given Deliverable and where this occurs, the functional specification will take precedence over the checklist
- There may be circumstances where the guidelines cannot be adhered to because of limitations in the software being used to implement a Deliverable. Where this occurs, justification for failure to meet the accessibility requirements should be supplied
- The accessibility requirements detailed in this document do not apply to ‘out of the box’, packaging or installation procedures

2 THE 10 HIGH LEVEL ACCESSIBILITY REQUIREMENTS

The following high level requirements were derived from an analysis of existing publicly available documents that pertain to the accessibility of desktop and web applications; for justifications and explanations, see *Design Guidance – Accessibility Principles* **{R1}**:

1. Support standard system size, colour, font, input settings, and accessibility options.
2. Enable programmatic access to user interface elements and text.
3. Provide keyboard access to all features.
4. Expose the location of the keyboard focus.
5. Provide equivalents for non-text elements.
6. Do not rely exclusively on a single perceptual capability to convey information.
7. Avoid flashing elements.
8. Enable user control of timed information presentation and responses.
9. Ensure consistency between interface elements and display items.
10. Create accessible documentation about accessibility features.

The above ten requirements must be met in order to achieve accessibility, and it is further recommended that:

- User-customisation at the application/interface level is included whenever beneficial for usability and accessibility
- Attention is applied at the design and implementation stages to the navigational flow and contextual feedback mechanisms – these must be logical and, where needed, modifications made to make them more accessible
- Applications, interface components and display items are user-tested with participants that include users of adaptive technology. This is important if accessibility is to be truly realised. User testing may become a requirement in the future.

In the following section the above ten high level accessibility requirements are expanded into a number of specific checkpoints.

3 ACCESSIBILITY CHECKPOINTS

3.1 Introduction

This section provides details of the accessibility requirements, checkpoints, and testing methods for each checkpoint. A table listing all of the requirements with their associated checkpoints is first provided, followed by detail for each checkpoint.

Checkpoint details take the following form:

- **Requirement** – the overall principle and requirement for accessibility that should be met by the software. A single requirement may have several associated checkpoints.
- **Checkpoint** – specific checkpoints that relate to the overlying accessibility requirement.
- **Test Method** – a description of an appropriate testing methodology to check whether the overlying checkpoint has been met.
- **Pass Conditions** – the requirements necessary for the overlying checkpoint to be considered satisfied. For some checkpoints, an exclusion note is provided that offers guidance on special conditions that can be considered exempt from needing to meet the pass conditions.
- **Fail Conditions** – statements that, if found during the testing, lead to the checkpoint being considered to be failed.
- **Notes** – additional information concerning the scope of the checkpoint, the testing methodology and guidance on some aspects of implementation. The implementation notes should not be considered exhaustive or mandatory; they are provided to offer helpful assistance only.

3.2 Checklist of Checkpoints

Requirements and Associated Checkpoints		Pass	Fail	N/A
R 1	**Support standard system size, colour, font, input settings, and accessibility options**			
C 1.1	Respect system font, size and colour settings			
C 1.2	Ensure compatibility with the high contrast option			
C 1.3	Ensure that keyboard and other accessibility features still function			
R 2	**Enable programmatic access to user interface elements and text**			
C 2.1	All text must be exposed to adaptive technology			
C 2.2	Descriptive titles must be supplied for windows, dialogues, frames, objects and pages			
C 2.3	All form controls must be labelled, and the labels should be exposed to adaptive technologies in a logical manner			
C 2.4	Data table information should be exposed to adaptive technologies			
R 3	**Provide keyboard access to all features**			
C 3.1	All menus, menu items, controls and hotspots must be navigable and operable using the keyboard alone			
C 3.2	All mouse operations must be able to be performed using the keyboard, including selections, drag and drop, resizing, moving and scrolling			
C 3.3	Accesskeys should be implemented for all standard options in menus and dialogues, and their function should be made easily discoverable			
C 3.4	Provide a logical tab order			
R 4	**Expose the location of keyboard focus**			
C 4.1	Ensure that the location of keyboard focus is indicated visibly for all elements that can receive focus			
C 4.2	Ensure that the location of keyboard focus is exposed programmatically for all elements that can receive focus			
C 4.3	Selections in non-active windows should be maintained but distinguished from the active selection			
R 5	**Provide equivalents for non-text elements**			
C 5.1	Provide text equivalents for all images. The equivalents must convey the same information			
C 5.2	Provide equivalent alternatives to animations, video and audio material			
C 5.3	Use real text characters rather than images of text wherever possible			
R 6	**Do not rely exclusively on a single modality (sense or perceptual capability) to convey information**			
C 6.1	Do not rely exclusively on visual presentation features (for example, colour, formatting) to convey information about the state or input requirements of the interface, or as a basis for interpretation of data			
C 6.2	Do not rely exclusively on sound to convey information about the state of the interface			
C 6.3	Provide sufficient luminance, contrast and colour contrast to permit use by people with colour blindness or mild visual impairment without needing to modify the default display			

Requirements and Associated Checkpoints		Pass	Fail	N/A
R 7	**Avoid flashing elements**			
C 7.1	Do not cause features to blink or flash in the range 2Hz to 59Hz			
R 8	**Enable user control of timed responses and time limited information presentation**			
C 8.1	Ensure that users have sufficient time to provide necessary inputs			
C 8.2	Ensure that users have sufficient time to interpret supplied data			
R 9	**Ensure consistency between interface elements and display items**			
C 9.1	Ensure that information display items have a consistent form and adhere to standards where defined			
C 9.2	Ensure consistency of navigation and interactions across the application			
R 10	**Create accessible documentation about accessibility features**			
C 10.1	Document all accessibility features			
C 10.2	Ensure that the documentation itself is accessible and discoverable			

Table 1: A Checklist of All the High Level Accessibility Requirements with their Associated Checkpoints

3.3 Requirement 1: Support Standard System Size, Colour, Font, Input Settings, and Accessibility Options

Checkpoint 1.1	Respect System Font, Size and Colour Settings

Test Method	Use the following procedures to change the appearance of the interface and check visually that font, size and colour settings are respected. The procedures involve making changes to the Windows® XP system settings using the Control panel.
	■ Control panel > Display > Appearance [tab]:
	  ■ Change the colour scheme (for example, olive green)
	  ■ Change the font size (for example, Normal)
	■ Control panel > Display > Appearance [tab] > advanced:
	  ■ Change the size for the 'active window' border item
	  ■ Change the font for the 'menu' item

Pass Conditions	■ Fonts, sizes and colours change to reflect the schemes chosen for all items in the list from Control panel > display > appearance [tab] > advanced (see the list in Notes below) ■ Exclusion note: Text in dialogues is currently excluded from Windows XP
Fail Conditions	■ One or more of the items do not reflect the size, colour or font settings applied

Notes	■ This checkpoint will normally be passed by default if standard controls are used. Special attention should be paid to custom controls
	■ The items to be checked are:
	  ■ 3D Objects
	  ■ Active Title Bar
	  ■ Active Window Border
	  ■ Application Background
	  ■ Caption Buttons
	  ■ Desktop
	  ■ Icon
	  ■ Icon Spacing (horizontal)
	  ■ Icon Spacing (vertical)
	  ■ Inactive Title Bar
	  ■ Inactive Window Border
	  ■ Menu
	  ■ Message Box
	  ■ Palette Title
	  ■ Scrollbar
	  ■ Selected Items
	  ■ Tooltip
	  ■ Window
	■ S*ee Windows Accessibility Features*[2] **{R3}** for further information on specific implementation techniques.

Table 2: Checkpoint 1.1: Respect System Font, Size and Colour Settings

[2]MSDN: Windows Accessibility Features: http://msdn2.microsoft.com/en-us/library/ms695623.aspx

Checkpoint 1.2	Ensure Compatibility With the High Contrast Option
Test Method	Use the following procedure to set the Windows XP high contrast option and check visually whether this is reflected in user interface controls and client area content. ■ Control panel > Accessibility Options > Display [tab] ■ Check the 'Use high Contrast' checkbox and apply the high contrast scheme ■ Control panel > Accessibility Options > Display [tab] > Settings: ■ Test an additional high contrast appearance scheme (for example, High contrast white extra large) to ensure that the interface is not dependent on a particular scheme
Pass Conditions	■ Colours and fonts change to reflect the high contrast schemes chosen for all user interface controls and client area content ■ Exclusion note: The High Contrast requirements do not apply to certain application features where the use of colour is intrinsic and indispensable to the goal of the feature. Examples include: ■ Palettes or swatches where the user selects from a range of displayed colours. In this case, the application can display the colour but should provide a text description such as a name (light blue) or numeric value (RGB 0, 255, 255) ■ Animation, video, and graphic images where the content is available through other means
Fail Conditions	■ The high contrast schemes are not reflected in the user interface controls or client area content ■ Information on the screen becomes invisible (for example, black on black) ■ Information on the screen becomes so disrupted to become unreadable (for example, garbled and overlapping)
Notes	■ The shortcut key combination (left ALT + left SHIFT + PRINT SCREEN) may be used to quickly toggle the high contrast setting ■ This checkpoint will normally be passed by default if standard controls are used. Special attention should be paid to custom controls ■ For the chosen high contrast scheme you can see how Windows controls should appear by looking at the appearance graphic. Use the following procedure to open this dialogue: ■ Control panel > Display > Appearance [tab] ■ Web applications should also be readable without stylesheets and should be able to support user stylesheets

Table 3: Checkpoint 1.2: Ensure Compatibility With the High Contrast Option

Checkpoint 1.3	Ensure that Keyboard and Other Accessibility Features Still Function
Test Method	Use the following procedures to enable the Windows XP keyboard accessibility features and check manually that they still function. ■ Control panel > Accessibility Options ■ Under the keyboard tab for StickyKeys, FilterKeys and ToggleKeys, select 'Settings' and ensure that the 'Use shortcut' checkbox is checked ■ Additionally for FilterKeys, select 'Settings' and ensure the 'Ignore repeated keystrokes' checkbox is checked ■ Under the display tab, select 'Settings' and ensure that the "use shortcut" checkbox is checked to enable high contrast ■ Under the mouse tab, select 'Settings' and ensure that the "use shortcut" checkbox is checked ■ Return to the interface that you are testing and perform the following checks: ■ Enable StickyKeys by pressing the SHIFT key five times ▪ Navigate within the application and test whether simultaneous key press combinations can be activated using sequential key presses. For example, with StickyKeys enabled CTRL+P (print) can be performed using CTRL and then P rather than both together ■ Enable FilterKeys by holding down SHIFT for 8 seconds ▪ Go to a text entry area within the application and press and hold down a standard letter key (for example, the 'P' key). Check whether multiple letters are displayed on the screen. With FilterKeys enabled, only one letter should be displayed ■ Enable ToggleKeys by holding down NUMLOCK for 5 seconds ▪ Press the Cap Lock, Num Lock and Scroll Lock keys, and listen for an audio tone (beep). With ToggleKeys enabled, a beep should sound when the keys are pressed ■ Check the High Contrast keyboard shortcut is functional by pressing left ALT + left SHIFT + PRINT SCREEN. The display should change to reflect the setting (see checkpoint 1.2) ■ Enable MouseKeys by pressing left ALT + left SHIFT + NUMLOCK. Use the arrow keys to see whether the mouse pointer moves on the screen. With MouseKeys enabled, the arrow keys should cause the mouse pointer to move
Pass Conditions	■ All 5 keyboard accessibility features function as detailed in the checks above
Fail Conditions	■ One or more of the keyboard accessibility features fail to work as expected
Notes	■ More information about how the keyboard accessibility features should function is detailed in the Help supplied with Windows XP

Table 4: Checkpoint 1.3: Ensure that Keyboard and other Accessibility Features Still Function

3.4 Requirement 2: Enable Programmatic Access to User Interface Elements and Text

Checkpoint 2.1	All Text Must Be Exposed to Adaptive Technology
Test Method	To test this checkpoint, one of the following screen readers is required: ■ JAWS® for Windows® ■ Window-Eyes™ Using a screen reader, navigate through the application to ensure that all text is read out audibly. If the screen reader fails to read a text element, inspect the code either manually or using an appropriate code inspector.
Pass Conditions	■ All text is capable of being read out using at least one of the screen readers. If not, the text is exposed properly as determined by code inspection (see Notes) ■ Exclusion note: it is conceivable that, for some UI elements, text is not exposed under normal circumstances. In this case, an accessible alternative that is capable of supplying the equivalent information contained in the text must be supplied

Fail Conditions	■ Any text is unavailable to both of the screen readers and, where this is the case, either of the following conditions is true:
	  ■ Text is not exposed properly as determined by code inspection
	  ■ There is no way to access an equivalent alternative with the above screen readers that presents the same information as contained in the text element

Notes	■ Standard mechanisms to send text through the operating system's graphics device interface (GDI) for display on the screen should be used wherever possible. Standard mechanisms include:
	  ■ The Windows Edit and Static Text controls
	  ■ The Microsoft RichEdit and HTML controls
	  ■ The Status control provided by Windows Common Controls
	  ■ The following APIs: TextOut, ExtTextOut, TabbedTextOut, DrawText, DrawTextEx, and PolyTextOut
	■ Programmatic access can be enabled by supporting the following:
	  ■ Windows UI Automation properties, control patterns, events, and logical tree structure
	  ■ Microsoft Active Accessibility (MSAA) IAccessible interface for all UI elements, or the ITextStore interface for textual content
	  ■ See *How to Provide Programmatic Access to UI Elements and Text*[3] **{R4}** for further guidance on fulfilling this checkpoint for non-web application
	■ For web applications it is important to code to standards such that pages will validate (for example, using http://validator.w3.org). At the present time, the minimum platform for running clinical applications remains to be defined. Until that time, care should be taken to avoid reliance on JavaScript or other technologies or plugins as the sole means of delivering information. This is necessary to achieve the required WAI-AA compliance

Table 5: Checkpoint 2.1: All Text Must Be Exposed to Adaptive Technology

[3]MSDN: Accessibility Best Practices – Enables Programmatic Access to all UI Elements and Text: http://msdn2.microsoft.com/en-us/library/aa350483.aspx

Checkpoint 2.2	Descriptive Titles Must be Supplied for Windows, Dialogues, Frames, Objects and Pages
Test Method	To test this checkpoint, one of the following screen readers is required: ■ JAWS for Windows ■ Window-Eyes Navigate through the application to check whether each window, frame, object, dialogue and page has an appropriately descriptive and diagnostic title. Ensure that this can be read out by the screen reader. If the screen reader fails to read a title, inspect the code either manually or using an appropriate code inspector.
Pass Conditions	■ All windows, dialogues, frames, objects and pages have a title that is exposed to the screen reader (or coded appropriately) ■ The title supplied is unique and informative about the purpose of the element to which it refers

Fail Conditions	■ Titles are missing for one or more windows, dialogues, frames, objects or pages
	■ One or more titles are not suitably descriptive to satisfy both the following conditions:
	  ■ Uniquely identify the element to which it refers – that is, it is suitable to differentiate between elements
	  ■ Usefully identify the element to which it refers – that is, it is diagnostic of the element to which it specifically relates

Notes	■ In some circumstances (for example, a frame in a web application) an additional description may be necessary if the purpose of the element, and how it relates to other elements, is not clear from the title alone

Table 6: Checkpoint 2.2: Descriptive Titles Must be Supplied for Windows, Dialogues, Frames, Objects and Pages

Checkpoint 2.3	All Form Controls Must be Labelled, and the Labels Should be Exposed to Adaptive Technologies in a Logical Manner
Test Method	To test this checkpoint, one of the following screen readers is required: ▪ JAWS for Windows ▪ Window-Eyes Navigate through the application to check whether each and every form control has a label. Check with the screen reader whether the label and form control are appropriately associated. For example, when in a text input control, the label read out is correct for that control, and should be read out first. If the screen reader fails to read a label, inspect the code either manually or using an appropriate code inspector.
Pass Conditions	▪ All form controls have an appropriate label that is exposed to the screen reader. If not, the text in the labels is exposed properly as determined by code inspection (see Notes) ▪ Labels are positioned properly such that the label is read out with the appropriate form control in the appropriate order (that is, label then control) ▪ Exclusion note: in some circumstances a separate label may not be required if the control is implicit in its function and the function is readily apparent when using a screen reader

Fail Conditions	▪ Labels are missing on one or more form controls, and the input or action required is not readily exposed to a screen reader (and not coded properly as determined by code inspection)
	▪ Labels are not associated with form controls such that they are not read out in a logical order to a screen reader (and not coded properly as determined by code inspection)
	▪ One or more labels are not suitably descriptive to satisfy both the following conditions:
	  ▪ Uniquely identify the control to which it refers – that is, it is suitable to differentiate between controls
	  ▪ Usefully identify the control to which it refers – that is, it is diagnostic of the input or action required of the associated control

Notes	▪ Normally, text labels should be placed immediately to the left or immediately above the control. Screen readers use proximity to identify labels if they are not exposed programmatically ▪ In Windows applications, check boxes and radio buttons have built in captions, whereas other controls do not (for example, lists and edit controls). Provide captions (static text) for controls that do not have captions. Place them appropriately (immediately left or above the control) and set the tab order of the caption control so that it immediately precedes the associated control ▪ In web applications, the control should be explicitly associated with its label using the LABEL element. In some circumstances the TITLE attribute can also be used ▪ See *Enable Programmatic Access to all UI Elements and Text*[4] **{R4}** for further guidance on fulfilling this checkpoint for non-web applications

Table 7: Checkpoint 2.3: All Form Controls Must be Labelled, and the Labels Should be Exposed to Adaptive Technologies in a Logical Manner

[4] MSDN: Accessibility Best Practices – Enable Programmatic Access to all UI Elements and Text: http://msdn2.microsoft.com/en-us/library/aa350483.aspx

Checkpoint 2.4	Data Table Information Should be Exposed to Adaptive Technologies
Test Method	To test this checkpoint, one of the following screen readers is required: ■ JAWS for Windows ■ Window-Eyes Navigate through the application to check whether tables have appropriate titles / captions and that the contents of all cells are exposed to the screen reader. If the screen reader fails to read a label, inspect the code either manually or using an appropriate code inspector. For web applications, additional tests should be performed to check whether row and column headings have been identified and appropriately associated with data cells, and that additional useful information is supplied (see Notes).
Pass Conditions	■ Data in tables is available to screen readers. If not, table information is exposed properly as determined by code inspection (see Notes) ■ It is clear from captions, titles or context what the data table contains ■ Where possible, column and row headers are identified (this is implementation specific – see Notes) ■ Where possible, cell contents are appropriately associated with column and row headers (this is implementation specific – see Notes)

Fail Conditions	■ Data in tables is not available to screen readers (and not coded properly as determined by code inspection)
	■ It is unclear or ambiguous what the contents of the data table are
	■ Where it is technically possible (and always for web applications), any of the following are not satisfied:
	  ■ Row and column headers are identified
	  ■ Cell contents are appropriately associated with column and row headers
	  ■ Additional information, such as summaries and captions, are supplied

Notes	■ For web applications, a check for whether row and column information is appropriately associated can be made navigating to a cell in the middle of the table and using the 'current cell' command Ctrl+Alt+numpad 5. Alternatively the table should be inspected for the presence of <TH> and associations using either SCOPE or HEADERS and ID ■ For web applications, useful information can be supplied regarding the layout and function of the table using the summary attribute ■ For Windows applications, MSAA does not provide specific support for tables. However, information on exposing table attributes using the MSAA IAccessible interface can be found in *Exposing Data Tables through Microsoft Active Accessibility*[5] **{R5}.** This, or other techniques appropriate to the underlying technology, should be used wherever possible

Table 8: Checkpoint 2.4: Data Table Information Should be Exposed to Adaptive Technologies

[5] MSDN: Exposing Data Tables through Microsoft Active Accessibility: http://msdn2.microsoft.com/en-us/library/ms971325.aspx

3.5 Requirement 3: Provide Keyboard Access to All Features

Checkpoint 3.1	All Menus, Menu Items, Controls and Hotpots Must be Navigable and Operable Using the Keyboard Alone
Test Method	Unplug the mouse and check that all the above elements are fully operable using the standard keys of the keyboard.
Pass Conditions	▪ All menus, menu items, controls and hotspots can be accessed and actioned by the keyboard without the need for a mouse or software that simulates a mouse, such as MouseKeys
Fail Conditions	▪ One or more of the elements cannot be accessed or actioned solely using the standard keyboard keys

Notes	▪ For toolbars, equivalent menu items must be provided, or direct access to the toolbar provided (preferably both)
	▪ It must be possible to switch between windows, panes and modeless dialogues using the keyboard
	▪ It should be possible to skip over groups of menu items rather than needing to repetitively 'tab' through all the items during navigation. In web applications, this should be explicitly included as a skip link
	▪ Care should be taken with drop down boxes where selection of an item automatically causes an action. It must be ensured that all the items can be navigated and selected using the keyboard. In a web application, it is preferable to have a separate 'submit' control
	▪ Standard keyboard actions should be supported where possible, for example:
	  ▪ Pressing the Tab key to navigate links, form controls, and objects
	  ▪ Pressing the Tab key to navigate through the content of each window
	  ▪ Pressing the arrow keys to navigate directionally
	  ▪ Typing the first few characters of a name to trigger an automatic search for names that begin with those characters
	▪ See *Guidelines for Keyboard User Interface Design*[6] **{R6}** for further guidance on supporting keyboard accessibility and adopting keyboard standards

Table 9: Checkpoint 3.1: All Menus, Menu Items, Controls and Hotpots Must be Navigable and Operable Using the Keyboard Alone

[6] MSDN: Guidelines for Keyboard User Interface Design: http://msdn2.microsoft.com/en-us/library/ms971323.aspx

Checkpoint 3.2	All Mouse Operations Must be Able to be Performed Using the Keyboard, Including Selections, Drag and Drop, Resizing, Moving and Scrolling
Test Method	Unplug the mouse and check that all mouse operations are fully operable using standard keys of the keyboard, including: ▪ Selection ▪ Drag and drop ▪ Resizing ▪ Moving ▪ Scrolling
Pass Conditions	▪ All mouse operations that are critical for functionality are achievable using the keyboard without the need to resort to simulation of the operation of a mouse using features such as MouseKeys ▪ Exclusion note: parts of an application that are inherently dependent on the fine level of control afforded by a mouse or other input device, such as a graphing tablet, are excluded. For example, a drawing program that requires **discrete pixel-by-pixel motor control** would be excluded
Fail Conditions	▪ One or more operations that are critical for functionality cannot be performed solely using standard keyboard keys
Notes	▪ It is not expected that a direct correspondence between mouse actions and keyboard actions is achieved; rather an equivalent method must be supplied that requires the keyboard alone. For example, where sizing of an element is achievable using click and drag from a mouse, it should also be possible to enter the required size and position in a form to achieve the same result. Similarly drag and drop operations should be able to be equivalently performed using keyboard operable cut and paste ▪ See *Guidelines for Keyboard User Interface Design*[7] **{R6}** for further guidance on supporting keyboard accessibility and adopting keyboard standards

Table 10: Checkpoint 3.2: All Mouse Operations Must be Able to be Performed Using the Keyboard, Including Selections, Drag and Drop, Resizing, Moving and Scrolling

[7] MSDN: Guidelines for Keyboard User Interface Design: http://msdn2.microsoft.com/en-us/library/ms971323.aspx

Checkpoint 3.3	Accesskeys Should be Implemented for All Standard Options in Menus and Dialogues, and their Function Should be Made Easily Discoverable
Test Method	Check that all options in menus and dialogues have a corresponding accesskey and that it works using the keyboard (Alt+key). Check that the accesskey is available to be visually presented to the user (usually as an underline). This may require you to perform the following action: ▪ Control panel > Accessibility Options > Keyboard [tab] ▪ Check the 'Show extra keyboard help in programs' checkbox.
Pass Conditions	▪ All standard options in menus have a corresponding unique accesskey ▪ All options / groups of options in a dialogue have a corresponding unique accesskey ▪ The appropriate accesskey for each option is available to be displayed visually ▪ Exclusion note: dynamically generated menus are excluded ▪ Exclusion note 2: web sites / applications are excluded due to conflicts with browser and adaptive technology key combinations (see Note below for further information on this point). However, care should be taken to group elements and options appropriately to assist keyboard operation ▪ Exclusion note 3: Form select boxes are excluded; although in dialogues, the select box should have an accesskey assigned to facilitate the user navigation to the select box
Fail Conditions	▪ One or more options in a Windows application menu does not have a unique accesskey ▪ One or more options / group of options in a Windows dialogue does not have a unique accesskey ▪ The corresponding accesskey is not available to be displayed visually to the user
Notes	▪ Ideally the accesskeys should be chosen to be those that are most memorable and intuitive, for example 'F' for file menu, 'E' for edit menu ▪ For dialogue boxes that present many options requiring input from the user, the options should be grouped logically and an accesskey provided for the group ▪ Accesskeys differ from keyboard shortcuts in that keyboard shortcuts are usually globally available throughout an application, whereas accesskeys are more context dependent. For example, Ctrl+S is a common keyboard shortcut for Save that is typically available. Whereas the accesskey 'S' activated through Alt+S changes depending on context. For example, with a typical file menu open, it may 'Save', whereas with a typical edit menu open, it may 'Paste Special' ▪ It is highly recommended that keyboard shortcuts be implemented for the most common tasks – this will also function to assist accessibility ▪ For web applications, the use of accesskeys is problematic due to conflicts with browser and adaptive technology key combinations

Table 11: Checkpoint 3.3: Accesskeys Should be Implemented for All Standard Options in Menus and Dialogues, and their Function Should be Made Easily Discoverable

Checkpoint 3.4	Provide a Logical Tab Order
Test Method	Use only the Tab key to navigate through the elements in application windows, panes and dialogues and assess whether the tab order corresponds to the logical visual order. Use Shift+Tab to check that this reverses the direction of navigation.
Pass Conditions	▪ A logical tab order is provided ▪ Shift+Tab is operable to reverse the direction of navigation provided by the Tab key
Fail Conditions	▪ One or more operations cannot be performed using the keyboard alone
Notes	▪ For western languages, the logical tab order is assumed to be left to right and top to bottom ▪ There is room for interpretation over what constitutes a logical order. The recommended way to assess this is to perform user testing that includes people dependent on the keyboard, such as people with mobility impairments who cannot use a mouse, and people who are blind and cannot use a mouse ▪ In particular, the tab order of forms should follow a logical manner that corresponds to what one would expect visually

Table 12: Checkpoint 3.4: Provide a Logical Tab Order

3.6 Requirement 4: Expose the Location of Keyboard Focus

Checkpoint 4.1	Ensure that the Location of Keyboard Focus is Indicated Visibly for All Elements that Can Receive Focus
Test Method	Navigate within the application using the Tab and arrow keys. Verify that there is a visual indication of the object that has focus.
Pass Conditions	▪ For all elements that can receive focus, the focus is communicated visually
Fail Conditions	▪ Visual indication of the focus is lost for one or more elements ▪ More than one keyboard focus is indicated at any one time

Notes	▪ This checkpoint will normally be passed by default if standard controls are used. Special attention should be paid to custom controls
	▪ Visual focus may be indicated using different methods, including:
	  ▪ Focus rectangles
	  ▪ Highlighting / shading
	  ▪ Insertion caret / cursor
	▪ Menus, dialogues and client input areas are all included for this checkpoint
	▪ Keyboard focus should be able to be indicated separately from selection to permit multiple or disjoint selection
	▪ This test may be carried out at the same time as the test for checkpoint 4.2, which uses a screen magnification device
	▪ See *Guidelines for Keyboard User Interface Design*[8] **{R6}** for further information on appropriate techniques to expose the keyboard focus:

Table 13: Checkpoint 4.1: Ensure that the Location of Keyboard Focus is Indicated Visibly for All Elements that Can Receive Focus

[8] MSDN: Guidelines for Keyboard User Interface Design: http://msdn2.microsoft.com/en-us/library/ms971323.aspx

Checkpoint 4.2	Ensure that the Location of Keyboard Focus is Exposed Programmatically for All Elements that Can Receive Focus

Test Method	Use the following procedure to enable the Windows XP magnifier features and check visually that the focus is displayed correctly:
	■ Launch 'magnify.exe' from the run dialogue (Start > Run), and use the following settings:
	  ■ Magnification level – 2
	  ■ Follow mouse cursor – unchecked
	  ■ Follow keyboard focus – checked
	  ■ Follow text editing – checked
	  ■ Show Magnifier – checked
	■ Enlarge the magnification target as appropriate for comfortable viewing (a third to a half of the vertical screen)
	■ Give focus to the application under test and navigate to each element using the tab and arrow keys. Verify that the item that has focus has become enlarged by the magnifier and is displayed on the screen

Pass Conditions	■ For all objects that can receive focus, the object is displayed magnified on the visible screen
Fail Conditions	■ On-screen visual display of the element under focus is lost for one or more elements
Notes	■ This checkpoint will normally be passed by default if standard controls are used. Special attention should be paid to custom controls ■ This test may be carried out at the same time as the test for checkpoint 4.1 ■ This test could also be carried out using commercially available magnification tools

Table 14: Checkpoint 4.2: Ensure that the Location of Keyboard Focus is Indicated Visibly for All Elements that Can Receive Focus

Checkpoint 4.3	Selections in Non-Active Windows Should be Maintained but Distinguished from the Active Selection
Test Method	Perform a visual check to ensure that it is clear from visual inspection to distinguish between 'active' selections and non-active selections
Pass Conditions	■ It is clear from visual inspection which selection is 'active'
Fail Conditions	■ It is unclear or ambiguous from visual inspection which selection is active
Notes	■ 'Greying-out'/dimming is commonly used but other techniques could be considered ■ An active selection may be considered to be the selection that would have any action immediately performed upon it. These are usually in the window that has the current keyboard focus. For example, when multiple text selections occur across multiple windows, the active selection is the text that would be removed by performing a 'cut' operation

Table 15: Checkpoint 4.3: Selections in Non-Active Windows Should be Maintained but Distinguished from the Active Selection

3.7 Requirement 5: Provide Equivalents for Non-Text Elements

Checkpoint 5.1	Provide Text Equivalents for All Images. The Equivalents Must Convey the Same Information
Test Method	To test this checkpoint, one of the following screen readers is required: ■ JAWS for Windows ■ Window-Eyes Navigate using a screen reader to all images in the application and listen to ensure that equivalent information is provided audibly.
Pass Conditions	■ Appropriate equivalent information that conveys the same information is provided to the screen reader for all images ■ Using a screen reader, it must be possible to gain the same quality of information conveyed through each image that supplies information ■ Using a screen reader, it must be possible to perform the same actions where the availability of actions are indicated by images

Fail Conditions	■ The use of an image prevents essential information being conveyed by a screen reader
	■ The use of an image prevents essential actions being performed by screen reader users
	■ One or more of the text equivalents is not suitably descriptive to satisfy both the following conditions:
	  ■ Uniquely identify the image to which it refers – that is, it is suitable to differentiate between different images
	  ■ Equivalently convey the information contained in the image – that is, the same quality of information is conveyed by the alternative description

Notes	■ There can be some ambiguity concerning what constitutes a text equivalent. Rather than a literal description of an image, the equivalent must convey the essential information or provide a functional description that gives the purpose of the image. Where ambiguity exists, the descriptions should be user-tested with people with the appropriate domain knowledge (for example, doctors). They should be presented with only the alternative and asked to perform an appropriate task based on the information the alternative conveyed ■ For non-essential graphical information, such as decoration, a literal description should not be provided. Instead, a NULL value should be given that would be ignored by assistive technologies such as screen readers ■ Where images provide redundant information that is supplied using real text in an accessible form, the image does not need to repeat the text. For example, this might be where an icon is presented to support a text label ■ Where an image is impossible to translate into text, for example information that is purely visual such as an X-ray image, a functional description should be given. In this case, the text supplied should identify the image as an X-ray (it is assumed that appropriate accessibility would be provided to the interpretation of the X-ray information) ■ For Windows applications, use the standard Windows tooltip control to apply a label to each image, or use an appropriate interface to expose the equivalent programmatically (for example MSAA or Automation) ■ Use of an appropriate object inspector for MSAA or Automation is recommended as a further check for this checkpoint – this can be used to ensure that the name and description of each label is appropriately provided ■ In a web environment, use the `alt` attribute to provide the text equivalent of each image and, where the image is complicated, use the `longdesc` attribute to link to a URL that contains an appropriately detailed description

Table 16: Checkpoint 5.1: Provide Text Equivalents for All Images. The Equivalents Must Convey the Same Information

Checkpoint 5.2	Provide Equivalent Alternatives to Animations, Video and Audio Material
Test Method	To test this checkpoint, one of the following screen readers is required: ■ JAWS for Windows ■ Window-Eyes Manually inspect the interface for the provision of equivalent alternatives, which may be: ■ Transcripts, or otherwise equivalent descriptions of video and audio material ■ Closed captions ■ Optional audio descriptions ■ Sign language alternatives to audio tracks Use the screen reader to ensure that the equivalent alternative is available. For example, closed captions must be able to be read out by the screen reader.
Pass Conditions	■ An appropriate alternative to animations, video and audio material is provided. These must be available for use by appropriate adaptive technologies ■ If the timing of the information supplied in the animation, video track or audio track is essential, the equivalent must be appropriately synchronised (using closed captions and audio descriptions)
Fail Conditions	■ The use of an animation, video track or audio track prevents essential material being conveyed while using a screen reader ■ The use of an animation, video track or audio track prevents essential material being conveyed that is only available as audio information
Notes	■ There can be some ambiguity concerning what constitutes an appropriate alternative for an animation, video track or audio track. The recommended way to assess this is to perform user testing that includes people with specific domain knowledge (for example, doctors). They should be presented with only the alternative and asked to perform an appropriate task based on the information the alternative conveyed ■ At the very least, a transcript or other equivalent description should be provided. Given more context concerning the possible roles of interface elements and the people that will use them, it will be possible to specify which method or methods must be used to present the equivalent ■ If multiple language versions of transcripts or captions are provided, the language should be referenced using an appropriate technique (for example a `lang` attribute for XML compliant technologies) ■ See also those checkpoints under requirement 6 – do not rely exclusively on a single modality (sense) to convey information ■ See *Captions and Audio Descriptions for PC Multimedia*[9] **{R7}** for further information on techniques for supplying alternatives for multimedia

Table 17: Checkpoint 5.2: Provide Equivalent Alternatives to Animations, Video and Audio Material

[9]MSDN: Captions and Audio Descriptions for PC Multimedia: http://msdn2.microsoft.com/en-us/library/ms971317.aspx

Checkpoint 5.3	Use Real Text Characters Rather than Images of Text Wherever Possible
Test Method	Perform a visual inspection to see if there are any images of text that could instead be presented as real text characters and, therefore, be directly accessible to modifications that increase accessibility (such as increased text size or high contrast). Enabling the high contrast setting (see checkpoint 1.2) can be used to facilitate the visual inspection. Images of text will not inherit the high contrast settings.
Pass Conditions	▪ No images that could be equally rendered as real text are present
Fail Conditions	▪ There are one or more images that could equally be rendered as real text and are therefore directly accessible
Notes	▪ Common violations of this checkpoint are section or group titles within application windows, or on web pages

Table 18: Checkpoint 5.3: Use Real Text Characters Rather than Images of Text Wherever Possible

3.8 Requirement 6: Do Not Rely Exclusively on a Single Modality (Sense or Perceptual Capability) to Convey Information

Checkpoint 6.1	Do Not Rely Exclusively on Visual Presentation Features (for example, Colour, Formatting) to Convey Information About the State or Input Requirements of the Interface, or as a Basis for Interpretation of Data
Test Method	Navigate through the application to identify the following: ■ Objects that change colour ■ Objects that change presentation style (such as font changes) Ensure that no information communicated by the interface is dependent *solely* on visual presentation / formatting. Use the tools listed above to alter the default presentation of information to assist with the inspection. The screen reader will read out the information independently of any visual style or formatting and the other tools will modify the default visual formatting.
Pass Conditions	■ All application information that is conveyed through visual presentation is also conveyed in a redundant manner. This will usually be through redundant text or other visual features (for example, marker icons) but could conceivably be some other appropriate perceptual modality. This applies to: ■ Information concerning the state of the application (for example, the current location) ■ Information concerning the requirements for input (for example, required fields) ■ Information necessary for interpretation of data (for example, highlighting new items in a list)
Fail Conditions	■ Information is conveyed using visual presentation features (for example, colour / formatting) alone. This information is not available when performing one or more of the following: ■ Listening with a screen reader ■ Viewing the interface in black and white ■ Viewing the interface with the high contrast setting
Notes	■ To assist with testing this checkpoint, the following tools are suggested: ■ A screen reader – Jaws for Windows or Window-Eyes ■ ColorDoctor – a visual filter tool that displays the screen in black and white ■ The high contrast setting in Windows XP (see checkpoint 1.2) ■ Common violations of this checkpoint include: ■ Highlighting the current location with a colour change ■ Identifying required fields in forms using colour or italics ■ Identifying errors on forms using colour changes to the text label ■ Providing information in lists or data tables by colour highlighting (for example, green = new) ■ Information content, rather than state of the application, is covered by the checkpoints under requirement 5. Equivalents for non-text elements need to be provided ■ This checkpoint does not imply that colour and/or formatting should not be used to highlight information, rather that it should not be the sole means of conveying the information. The common way to satisfy this checkpoint is to provide redundant (accessible) text or symbols, these include: ■ + (positive) ■ - (negative) ■ ( ) (negative) ■ * (flags a required item) ■ | (surrounds changes) ■ ? (help) ■ New (descriptive text) ■ The default for non-speech audio should be in the frequency band 500 Hz to 3000 Hz

Checkpoint 6.1	Do Not Rely Exclusively on Visual Presentation Features (for example, Colour, Formatting) to Convey Information About the State or Input Requirements of the Interface, or as a Basis for Interpretation of Data
	▪ For alerts, it is recommended that there should be at least 2 strong mid to low frequency components, one from each of the following ranges: ▪ 300 Hz to 730 Hz ▪ 500 Hz to 3000 Hz

Table 19: Checkpoint 6.1: Do Not Rely Exclusively on Visual Presentation Features (for example, Colour, Formatting) to Convey Information About the State or Input Requirements of the Interface, or as a Basis for Interpretation of Data

Checkpoint 6.2	Do Not Rely Exclusively on Sound to Convey Information About the State of the Interface
Test Method	Inspect the application to ensure that all information conveyed through sound has an appropriate redundant visual alternative.
Pass Conditions	▪ All information that is conveyed through sound is also available in a non-sound dependent manner
Fail Conditions	▪ Information is conveyed using visual presentation alone
Notes	▪ Common violations of this include alerts and notifications ▪ It may be possible in some instances to turn on and off the alternative visual presentation (for example, using ShowSounds) ▪ Information content, rather than state of the application, is covered by the checkpoints under requirement 5. Equivalents for non-text elements need to be provided

Table 20: Checkpoint 6.2: Do Not Rely Exclusively on Sound to Convey Information About the State of the Interface

Checkpoint 6.3	Provide Sufficient Luminance, Contrast and Colour Contrast to Permit Use by People with Colour Blindness or Mild Visual Impairment Without Needing to Modify the Default Display
Test Method	To assist with testing this checkpoint, the following tools are required: ■ ColorDoctor – a visual filter tool that displays the screen in ways that simulate different types of colour blindness. This tool also displays the underlying screen in black and white ■ Colour Contrast Analyser – a tool that can sample foreground and background colours to calculate colour brightness and colour difference Using ColorDoctor, navigate through the application while filtering the screens to simulate the three types of colour blindness: protanopia, deuteranopia, and tritanopia. Check for parts of the interface that become difficult to see. If background patterns are present, view with the black and white filter and judge whether the foreground information is difficult to read or interpret. Use the Colour Contrast Analyser to sample foreground and background colour combinations of user interface items that provide information or functionality. Check the results to ensure they provide sufficient colour brightness (>125) and colour difference (>400).

Pass Conditions	■ Information important to the function or use of the application remains visual under all 3 different views that simulate colour blindness
	■ Sufficient colour brightness of foreground and background is supplied according to the following minimum thresholds:
	  ■ Colour brightness – to be above 125
	  ■ Colour difference – to be above 400. Note that this value is different from the threshold value used by the tool (500). The lower figure is that recommended by Hewlett Packard using the same algorithm
	■ No complex backgrounds are present that disturb reading

Fail Conditions	■ Information important to the function or use of the application is lost visually under one or more of the three different views that simulate colour blindness ■ Colour brightness is below 125 (although see Note about large items) ■ Colour difference is below 400 (although see Note about large items)

Notes	■ There is no definitive algorithm that gives totally unambiguous results because of interactions between spatial frequency (size) and the human brain contrast sensitivity function (modulation transfer function). This means that, in general, large items (such as 20 pt text) can be seen at lower contrasts than small items (such as 8pt text). This needs to be taken into account before failing an element on this checkpoint. However, for text sizes that are normally present in applications (8pt or 9pt), the algorithms are appropriate and the thresholds should be respected
	■ The following colour combinations should generally be avoided as they cause problems for people who are colour blind:
	  ■ Green/red
	  ■ Green/blue
	  ■ Red/brown
	  ■ White/light green
	■ With high contrast set, images and other complex backgrounds from behind text should be removed to enhance readability (see also checkpoint 1.2)

Table 21: Checkpoint 6.3: Provide Sufficient Luminance, Contrast and Colour Contrast to Permit Use by People with Colour Blindness or Mild Visual Impairment Without Needing to Modify the Default Display

3.9 Requirement 7: Avoid Flashing Elements

Checkpoint 7.1	Do Not Cause Features to Blink or Flash in the Range 2Hz to 59Hz
Test Method	Inspect the application to locate elements or features that blink or flash. If present, assess the frequency of the blinking or flashing.
Pass Conditions	▪ The application contains no blinking or flashing elements ▪ The application contains blinking or flashing elements but the frequency of flashing is outside the 2Hz to 59Hz (cycles per second) range ▪ Exclusion note: If it is absolutely necessary for parts of the essential function of an application to contain elements that flash within the 2Hz to 59Hz range, it must be possible for the user to turn off the flashing element. In addition, prior warning must be supplied in an appropriate form so that viewers who may be sensitive to photosensitive epileptic seizures are not exposed to the dangerous stimulus accidentally
Fail Conditions	▪ The application contains blinking or flashing elements inside the 2Hz to 59Hz (cycles per second) range
Notes	▪ In animations, smooth transitions should be supplied to avoid introducing flashing or flicker ▪ It may be necessary to use an electronic tester to determine the exact frequency of flicker but, as a rule of thumb, if the flicker/flash rate is above 2 cycles per second and you can detect it visually, it will fall below 59Hz and will consequently fail this checkpoint ▪ Peak sensitivity for people prone to photosensitive epilepsy is 20Hz – this should be avoided at all costs

Table 22: Checkpoint 7.1: Do Not Cause Features to Blink or Flash in the Range 2Hz to 59Hz

3.10 Requirement 8: Enable User Control of Timed Responses and Time Limited Information Presentation

Checkpoint 8.1	Ensure that Users Have Sufficient Time to Provide Necessary Inputs
Test Method	Inspect the application to locate any functions, forms or dialogues that require input from the user. Assess whether a 'time-out' is in operation and whether there is a method of gaining sufficient time to perform the required actions without initiating a time-out.

Pass Conditions	▪ No time-outs are in operation on any function, form or dialogue that requires user input
	▪ If time-outs are in operation on any function, form or dialogue, one of the following conditions must be satisfied:
	  ▪ The user is able to request additional time to provide input without loss of data
	  ▪ The time allowed is sufficiently long that a time-out will not be initiated that causes loss of data before all users, including those with disabilities, are able to supply the required data

Fail Conditions	▪ Insufficient time is available for all users, including those with disabilities, to supply the required data before initiating a time-out. A fail is deemed to occur if the time-out results in either of the following conditions:
	  ▪ Data loss causing the requirement for repeating inputs that were already entered
	  ▪ A significant extra burden required to continue the entry of input. For example, by causing repeated navigation to the location of an input form

Notes	▪ The amount of time necessary for all users, including those with disabilities, to supply the required data before initiating a time-out, is highly context specific. The time will vary depending on the purpose of the input and the capabilities and experience of those entering the data. This will need to be determined individually for each input component, and will ideally be informed through user testing. Until this information is available, a reasonable rule of thumb[10] is that at least 10 times the average estimated response time for each activity should be allowed

Table 23: Checkpoint 8.1: Ensure that Users Have Sufficient Time to Provide Necessary Inputs

[10]The rule of thumb was suggested in the *Irish National Disability Authority IT Accessibility Guidelines: Accessibility Guidelines for Application Software:* http://accessit.nda.ie/technologyindex_4.html **{R8}**

Checkpoint 8.2	Ensure that Users Have Sufficient Time to Interpret Supplied Data
Test Method	Inspect the application to determine if a 'time-out' is in operation on any information display windows or panes. Assess whether there is a method of gaining additional time without initiating a time-out. Locate any discrete parts of the interface that provide information in the following forms: ■ 'Dynamically', for example in a marquee ■ 'Transiently', for example in a temporary alert or notification Determine if there is a means for controlling the presentation of the data in order to present it under user control, statically or for a user determined duration.

Pass Conditions	■ No time-outs, or dynamic or transient presentation of data is in operation
	■ If time-outs are in operation on any information displayed, or if information is presented transiently, one of the following conditions must be satisfied:
	  ■ The user is able to request additional time to permit interpretation of the data
	  ■ The time allowed is sufficiently long that a time-out will not be initiated before all users, including those with disabilities, are able to interpret data
	■ If information is displayed dynamically, one of the following conditions must be satisfied:
	  ■ The user is able to control the presentation of the data, for example a facility is provided that allows the user to stop a scrolling presentation of data (for example in a marquee)
	  ■ An alternative equivalent static representation of the information is provided that is not time dependent

Fail Conditions	■ Insufficient time is available for all users, including those with disabilities, to interpret data before initiating a time-out or loss of transient information. A fail is deemed to occur if the time-out results in either of the following conditions:
	  ■ A significant extra burden required to continue with interpretation of the data. For example, by causing repeated or convoluted navigation to the location of the data display
	  ■ Loss of transient information either:
	  ▪ Completely – because there is no way to recover it
	  ▪ Effectively – because the steps to recover it are convoluted or not easily discoverable, or because the time taken to recover the information renders it no longer valid

Notes	■ Even more so than for checkpoint 8.1, the time dependency for data interpretation is highly context specific. The time will vary depending on the purpose of the data, the context of use and the capabilities and experience of those interpreting it. This will need to be determined individually for each data display component, and will ideally be informed through user testing. Until this information is available, a reasonable rule of thumb is that at least 10 times the average estimated time required for data interpretation should be allowed

Table 24: Checkpoint 8.2: Ensure that Users Have Sufficient Time to Interpret Supplied Data

3.11 Requirement 9: Ensure Consistency Between Interface Elements and Display Items

Checkpoint 9.1	Ensure that Information Display Items Have a Consistent Form and Adhere to Standards Where Defined
Test Method	Perform a visual inspection across the application to determine consistency and adherence to any defined standards.
Pass Conditions	■ All of the following conditions should be met: ■ Information display items are displayed in a consistent form across the entire application, which includes: ▪ The data display format of the element itself, for example if an item takes the form NNN NNN NNN, this format should be universally used wherever it occurs (unless alternative formats are specifically defined) ▪ Any labels used to refer to data display items, for example if ABCDEF is always used to refer to a particular data element, this should be universally used wherever it occurs ▪ Any labels or terminology used in menus ▪ Any graphics used to denote information (such as alerts) or functions ■ The relationship between individual information items and groups of information items are displayed consistently across the application with respect to: ▪ Their visual spatial relationships ▪ Their relationship in the tab or readout order ■ Where a standard exists, the standard defined form of the display item should be used
Fail Conditions	■ Information display items are inconsistently displayed ■ Available defined standards are not used ■ The meaning of a graphic or icon changes ■ The same item appears on the same page but has different results. For example, multiple 'click here' links on a web page
Notes	■ A number of standards for data display elements have been defined for clinical applications, which include: ■ *Design Guidance – Address Information Display* **{R9}** ■ *Design Guidance – Patient Identification Number Display* **{R10}** ■ *Design Guidance – Telephone Number Display* **{R11}** ■ *Design Guidance – Gender and Sex Display* **{R12}** ■ *Design Guidance – Date Display* **{R13}** ■ *Design Guidance – Time Display* **{R14}** ■ There may be special reasons for inconsistency of spatial relationships between elements, but these should be deliberate and logical

Table 25: Checkpoint 9.1: Ensure that Information Display Items Have a Consistent Form and Adhere to Standards Where Defined

Checkpoint 9.2	Ensure Consistency of Navigation and Interactions Across the Application
Test Method	Navigate within the application to ensure consistency of navigation using a mouse. Repeat the same steps using the keyboard to ensure consistency of tab order and key presses across the application.
Pass Conditions	▪ All of the following conditions should be met: ▪ Menus must appear in the same form and the same or similar spatial location across the application ▪ The methods required to use menus or menu items must be consistent, for example click, double-click, drag ▪ Items requiring user input should be of consistent appearance and the interaction model should be consistent. For example, forms should be consistent in design and function ▪ Tab order should be consistent (and logical, see checkpoint 3.4) across different views within the application ▪ Where a standard exists for keyboard interaction, the standard should be used; it should be noted that there is currently no standard defined for clinical applications, although standards for typical keyboard behaviour within Windows applications are available
Fail Conditions	▪ Any of the above pass criteria are not met, rendering the navigation and interaction modes inconsistent across the application
Notes	▪ Use of standard items such as standard display items and wizards aid consistency ▪ Use of a standard template or set of templates for web applications aids consistency ▪ Standard navigation mechanisms such as toolbars, wizards and navigation bars should be used wherever possible ▪ For web applications, pages should be consistently structured using heading levels to facilitate navigation (and usability)

Table 26: Checkpoint 9.2: Ensure Consistency of Navigation and Interactions Across the Application

3.12 Requirement 10: Create Accessible Documentation About Accessibility Features

Checkpoint 10.1	Document All Accessibility Features
Test Method	Manually inspect to ensure that all special accessibility features are documented.
Pass Conditions	▪ All accessibility features are specifically documented: ▪ Accesskeys and keyboard shortcuts ▪ Special keyboard operations that facilitate accessibility ▪ Any special accessibility features, options or workarounds
Fail Conditions	▪ Specific information is missing concerning one or more of the accessibility features
Notes	▪ Standard customisation options that enable the user to set their own font size and colour, colour options, sound options and element size should be covered in operating system documentation and do not need to be repeated ▪ Information about where to find accessible alternative formats should be included with any printed documentation ▪ Keyboard documentation for any task should be easy to find when it is needed. The user should be able to find the keyboard instructions with no more than one extra step than is required to get the mouse instructions. For example, can the user find the keyboard instructions by searching for "accessibility" or "keyboard shortcut"? If space is not available in the primary documentation, describe the keyboard interface elsewhere, and then cross-reference it ▪ Each procedural Help topic should describe both the mouse and the keyboard technique. If a Help topic can include only the mouse technique, the topic should have a link to a specific Help topic describing the associated keyboard technique. Stand-alone keyboard references are valuable as long as they supplement instructions in the procedural Help. However, they should not be used alone, because they prevent the user from easily finding instructions for the task at hand ▪ It is recommended that you include real-world examples that show users how to do things in an accessible way by using the accessibility features of your product ▪ The Help text itself should be as concise and informative as possible, with abbreviations and acronyms spelled out on first mention, and all new terminology defined on first mention

Table 27: Checkpoint 10.1: Document All Accessibility Features

Checkpoint 10.2	Ensure that the Documentation Itself is Accessible and Discoverable
Test Method	Manually check to ensure the documentation itself passes the above nine requirements and their associated checkpoints.
Pass Conditions	▪ The documentation passes all the relevant checkpoints to make it accessible ▪ The documentation passes relevant guidelines depending on the way it is delivered. For example, it may require extra validation against current web guidelines if the documentation is provided online in web format
Fail Conditions	▪ The documentation fails to be accessible to the checkpoints contained in this document ▪ The documentation fails to be validated against relevant guidelines pertaining to the specific form in which it is supplied
Notes	▪ Web format Help and documentation should be validated against the *Web Content Accessibility Guidelines*[11] **{R2}**

Table 28: Checkpoint 10.2: Ensure that the Documentation Itself is Accessible and Discoverable

[11] World Wide Web Consortium (W3C) Web Content Accessibility Guidelines (WCAG): http://www.w3.org/TR/WAI-WEBCONTENT/

4 Document Information

4.1 Terms and Abbreviations

Abbreviation	Definition
API	Application Programming Interface
GDI	Graphics Device Interface
ISO	International Organization for Standardization
MSAA	Microsoft Active Accessibility
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative

Table 29: Terms and Abbreviations

4.2 Nomenclature

This section shows how to interpret the different styles used in this document to denote various types of information.

All content subject to completion, agreement or verification is denoted with highlighting.

4.2.1 Body Text

Text	Style
Code	`Monospace`
Script	
Other markup languages	
Interface dialog names	**Bold**
Field names	
Controls	
Folder names	Title Case
File names	

Table 30: Body Text Styles

4.2.2 Cross References

Reference	Style
Current document – sections	Section number only
Current document – figures/tables	Caption number only
Other project documents	*Italics* and possibly a footnote
Publicly available documents	*Italics* with a footnote
External Web-based content	*Italics* and a hyperlinked footnote

Table 31: Cross Reference Styles

4.3 References

Reference	Document	Version
R1.	Design Guidance – Accessibility Principles	1.0.0.0
R2.	World Wide Web Consortium (W3C) Web Content Accessibility Guidelines 1.0 (WCAG) http://www.w3.org/TR/WAI-WEBCONTENT	
R3.	MSDN: Windows Accessibility Features: http://msdn2.microsoft.com/en-us/library/ms695623.aspxi	
R4.	MSDN: Accessibility Best Practices – Enables Programmatic Access to all UI Elements and Text: t http://msdn2.microsoft.com/en-us/library/aa350483.aspx	
R5.	MSDN: Exposing Data Tables through Microsoft Accessibility http://msdn2.microsoft.com/en-us/library/ms971325.aspx	
R6.	MSDN: Guidelines for Keyboard User Interface Design http://msdn2.microsoft.com/en-us/library/ms971323.aspx	
R7.	MSDN: Captions and Audio Descriptions for PC Multimedia http://msdn2.microsoft.com/en-us/library/ms971317.aspx	
R8.	Irish National Disability Authority IT Accessibility Guidelines: Accessibility Guidelines for Application Software http://accessit.nda.ie/technologyindex_4.html	
R9.	Design Guidance – Address Information Display	1.0.0.0
R10.	Design Guidance – Patient Identification Number Display	1.0.0.0
R11.	Design Guidance – Telephone Number Display	1.0.0.0
R12.	Design Guidance – Gender and Sex Display	1.0.0.0
R13.	Design Guidance – Date Display	1.0.0.0
R14.	Design Guidance – Time Display	1.0.0.0

Table 32: References

APPENDIX A TOOLS FOR TESTING

Name of Tool	Source URL
Jaws® for Windows®	http://www.freedomscientific.com/
	Notes: Currently version 7.0. A widely used screen reader produced by Freedom Scientific. An evaluation version is available for download.
Window-Eyes™	http://www.gwmicro.com/
	Notes: Currently version 6.1. A widely used screen reader produced by GW Micro. An evaluation version is available for download.
ColorDoctor®	http://design.fujitsu.com/en/universal/assistance/colordoctor/
	Notes: Currently version 2.01. A screen visualisation tool provided by Fujitsu. ColorDoctor filters screen content into black and white or simulates 3 forms of colour blindness.
Colour Contrast Analyser	http://www.visionaustralia.org.au/info.aspx?page=628
	Notes: Currently version 1.1. A tool for checking foreground and background colour combinations to determine if they provide good colour visibility. Uses algorithms supported by the W3C. The algorithms calculate a value for colour brightness and colour difference.
Microsoft® Active Accessibility® (MSAA) inspection tools	http://www.microsoft.com/downloads/details.aspx?familyid=3755582A-A707-460A-BF21-1373316E13F0&displaylang=en
	Notes: Tools provided as part of the Microsoft Active Accessibility 2.0 SDK. These tools can be used to inspect MSAA implementations.
MSAA Verify	http://www.codeplex.com/MsaaVerify
	Notes: A tool that you can point at an existing control to verify that it has implemented the IAccessible interface and specified nine MSAA properties for each of 10 of the possible RoleTypes the control implements.

Table 33: Tools for Assistance in Accessibility Testing