



**Web-Based Application Development  
Methodology for Section 508  
Compliance**

*February 28, 2002*

*Version 1.0*

CONTENTS

INTRODUCTION ..... 1

SCOPE ..... 3

DELIMITATIONS..... 5

BACKGROUND ..... 6

**What is Section 508? 6**

**What is the Purpose of Section 508? 6**

**How is Section 508 Implemented? 7**

USER COMMUNITY ..... 8

**Who Is the “User Community” of Government IT? 8**

**Who Was the “User” Before Section 508? 8**

**Who Are the “Users” Now? 8**

TECHNICAL STANDARDS ..... 10

**Functional Requirements 10**

**Appearance Requirements 11**

DEVELOPMENT LIFECYCLE..... 13

**Project Definition 13**

**Requirements Analysis 13**

**Renovating Existing Web-Based Applications 14**

**Building Accessible Web-Based Applications 14**

**Web-Based Application Testing 15**

SUMMARY ..... 18

APPENDIX A – TECHNICAL STANDARDS ..... 19

**Functional Requirements 19**

    Executing Function from Keyboard 19

    Accessibility Features 22

    Skip Links 24

    Input Focus 28

    Timed Response 30

    Scripts 33

    Bitmap Images 37

    Textural Information 40

**Appearance Requirements 43**

    Frames 43

    User Interface Element 46

    Color (Coding) 49

Color and Contrast Settings	51
Multimedia	52
Animation	58
Flicker or Blinking Text	60
Tables	61
Electronic Forms	65
Client-side Image Maps	69

APPENDIX B – SECTION 508 COMPLIANCE CHECKLIST FOR WEB-BASED APPLICATIONS .....	70
---	----

## INTRODUCTION

This is a 'living' document that examines the application of the Electronic and Information Technology (EIT) Accessibility Standards of Section 508 of the Rehabilitation Act Amendments of 1998 as they apply to web-based application development. This legislation requires that information delivered via Electronic and Information Technology (EIT) must be made accessible to disabled users in the Federal Government as well as the general public.

Subsequently, this development methodology is intended to be a starting point for incorporating accessibility into software design based on the Section 508 legislation and the Guide to Section 508 Standards for Electronic and Information Technology developed by the Federal Access Board. As new information becomes available and experience gained in the definition, interpretation, and application of accessible technology standards, it is expected that this methodology will be refined and expanded. The introduction of new software technology, programming languages, and systems integration technologies as well as advancements in assistive/adaptive technologies will also help to redefine and improve this methodology. It should further be noted that at the time of this document's publication, the Federal Access Board was revising their current Guide to the Section 508 Standards for Electronic and Information Technology and anticipated offering new training programs for developers of EIT in the near future.

The development methodology has been designed to produce web-based software applications that are (1) compliant with the Section 508 Technical Standards in effect at the time of the analysis, and, (2) accessible to disabled users and users of assistive technology.

To attain compliance with Section 508 technical accessibility standards requires a multi-phased approach:

- systems managers, software developers, government customers and contractors must be educated regarding Section 508 and advised in the correct application of the standards as they apply to NASA Headquarters
- existing applications must be tested and evaluated for Section 508 compliance
- applications currently in development must be re-evaluated to assure compliance in the delivered software products

The approach for making an application compliant with Section 508 will vary depending on the application's state of development (i.e., renovation of existing web-based applications or new development). Incorporating Section 508 development standards at the start of the development lifecycle will reduce labor and cost significantly as compared to the cost and labor of modifying an existing application's coding.

There are two significant coding issues facing the renovation process for existing applications at NASA Headquarters (HQ).

(1) Browser support. Since Netscape does not support accessibility coding, those applications developed for the Netscape browser must be re-coded to function correctly in the Microsoft Internet Explorer (IE) browser. These browsers do not support HTML, JavaScript or other DHTML languages the same way. In the process of re-coding the application for IE, the accessibility attributes should be added at the same time to reduce development costs.

(2) Application design. All web-based applications developed prior to June 21, 2001, were designed for a user with ideal eyesight, mobility and cognitive functions. The developer's ability to display large quantities of information within the confines of a browser's window has increased tremendously with advancements in technology. Information that would take many pages of paper to display can now be viewed within one browser window with the assistance of compound tables, JavaScript pop-up windows, hyperlink references and animations. Some content may be made accessible with modification to existing code, however much of the content may require a re-work of the user interface design (GUI) to implement an accessible presentation of the information. This revised content presentation may take the form of a new page design or the addition of an 'accessible' page set for disabled users.

This document provides a development framework that will support successful renovation of existing software applications and the development of new accessible software products.

This framework consists of:

- an explanation of Section 508
- an understanding of the design requirements of Section 508 and recommendations for implementing them
- a methodology for incorporating Section 508 requirements into the software development lifecycle to assure compliance
- tools to assist accessible development, e.g.,
  - Compliance Evaluation Checklist
  - Application Guide of Technical Standards for Accessibility
  - Online technical support resources for accessible development

Software applications analyzed during the production of this document were a combination of ColdFusion-based web applications with database application support.

*Note: It is recommended that hard copies of this document be printed on a color printer due to the treatment of accessibility features in Appendix A – Technical Standards Table.*

## SCOPE

This document provides a methodology for implementing the Section 508 Electronic and Information Technology (EIT) Accessibility Standards with regard to NASA HQ's current web-based applications. These web-based applications employ a combination of ColdFusion-generated HTML interface front-ends with database driven back-ends supported by programs such as Oracle. This combination of application components does not lend itself wholly to either the Section 508 1194.22 Web page Standards or the 1194.21 Software Application Standards. They are a combination of both Standards, with some of each applying. Thus, both sets of standards were combined to produce this methodology. This document identifies which standards are applicable to web-based applications and provides a guide for applying them. It further identifies the application of Section 508 requirements to both new and existing software applications.

Portions of the following applications were tested and modified to complete this analysis.

- IWMS
- RMRS
- KIC 4.2
- PSRS

Microsoft Internet Explorer (IE) 5.5 was used as the baseline browser in this analysis because of IE's support for accessibility functions for the following reasons:

- IE integrates and utilizes the accessibility features built in to the Windows operating environment. This complies with Section 508: Standard 1194.21 (b)
- IE employs user configurable accessibility features that support and integrate well with assistive technology. For example, the IE browser interface is developed to be accessible to assistive technology. The menu commands, user preference configuration menu, help menu and support programs -- such as notepad -- are all accessible to screen reader technology and the like
- Microsoft has made a commitment to support and expand their products support of assistive technology. This commitment has enabled developers to develop applications with improved performance and accessibility.
- Microsoft offers a Voluntary Product Accessibility Template (VPAT) or description of the accessibility features in the Internet Explorer web browser.

Netscape 4.7 was tested and proved not to be an accessible product in itself.

- Netscape does not offer a VPAT describing their product's accessibility features.
- Some Netscape menu items were not accessible or identified to assistive technology.

- Netscape does not support the accessibility features of the Windows OS and attempts to override the user selected font size and type.
- Netscape does not provide consistent support for the accessible HTML code elements available in HTML 4.0.
- Netscape does not integrate well with the Microsoft Active Accessibility layer which provides communication for many assistive software products by making on-screen text information available to various assistive technologies such as speech synthesizing software and refreshable Braille displays.
- Netscape's latest browser version 6 does not claim to incorporate support for assistive technology.

## DELIMITATIONS

The technical analysis was performed with the following recognized limitations.

- Due to funding and associated resource constraints, only the Windows-based platform perspective was examined.
- Renovation techniques developed during this analysis were a product of the manipulation of code samples rather than entire applications. Given the interdependency of code modules, the process of renovating an entire application may require significantly more work than the modification of selected code samples.

Therefore, this methodology should not be considered inclusive of all methods, techniques, and technologies available to assist in the development of web-based applications consistent with Section 508 requirements.

## BACKGROUND

### What is Section 508?

Section 508 of the Rehabilitation Act Amendments of 1998 requires, effective June 21, 2001, that Electronic and Information Technology (EIT) procured by the Federal Government must comply with the accessibility standards as defined by Section 508.

- These accessibility standards were developed by the Federal Access Board to provide a set of guidelines defining EIT accessibility. When developing these guidelines, the Federal Access Board consulted representatives of leading commercial software vendors, developers of assistive technology products, government agency representatives, and representatives of various organizations representing people with disabilities. The resulting guidelines have been finalized and published as the Section 508 Technical Standards.

The following is an excerpt from Section 508 of the Rehabilitation Act, Electronic and Information Technology Accessibility Standards:

***“SUMMARY: The Architectural and Transportation Barriers Compliance Board (Access Board) is issuing final accessibility standards for electronic and information technology covered by Section 508 of the Rehabilitation Act Amendments of 1998. Section 508 requires the Access Board to publish standards setting forth a definition of electronic and information technology and the technical and functional performance criteria necessary for such technology to comply with Section 508. Section 508 requires that when Federal agencies develop, procure, maintain, or use electronic and information technology, they shall ensure that the electronic and information technology allows Federal employees with disabilities to have access to and use of information and data that is comparable to the access to and use of information and data by Federal employees who are not individuals with disabilities, unless an undue burden would be imposed on the agency. Section 508 also requires that individuals with disabilities, who are members of the public seeking information or services from a Federal agency, have access to and use of information and data that is comparable to that provided to the public who are not individuals with disabilities, unless an undue burden would be imposed on the agency.”\****

*\*Architectural and Transportation Barriers Compliance Board 36 CFR Part 1194*

### What is the Purpose of Section 508?

Section 508 was adopted to remove the barriers to EIT that have been created as a result of inaccessible design for disabled users in the Federal Government and the general public. Therefore, it is incumbent on software developers to ensure that the accessibility standards are incorporated into all new software delivered to the Federal Government.

## How is Section 508 Implemented?

The implementation of Section 508 is performed through the regulation of Government procurement as defined by the Federal Acquisition Regulations (FAR). Government agencies are required under Section 508 to purchase the most 'accessible' answer to their IT needs.

The FAR has mandated the implementation of the Section 508 Standards in Subpart 39.2.

*"When acquiring EIT, agencies must ensure that:*

*(1) Federal employees with disabilities have access to and use of information and data that is comparable to the access and use by Federal employees who are not individuals with disabilities; and*

*(2) Members of the public with disabilities seeking information or services from an agency have access to and use of information and data that is comparable to the access to and use of information and data by members of the public who are not individuals with disabilities" \**

*\*From the Federal Acquisition Regulations Subpart 39.201*

This regulation has two distinct impacts. First government agencies have been given additional criteria by which they must consider all IT procurements. They must determine if their purchase is the 'best' financial answer to their IT needs as well as the best accessible answer as defined in the Section 508 Standards. Second, all vendors of IT products and services have been given a new implicit requirement defining their support of government agencies.

Consequently, the vendor must compete by providing not only the best answer to government's stated IT needs but must also provide IT solutions that are accessible. In addition, the vendor should provide the information necessary for government procurement agents to determine their product's accessibility merits as defined by the Section 508 Standards.

## USER COMMUNITY

### Who Is the “User Community” of Government IT?

The users of NASA HQ’s IT resources are NASA HQ’s employees and the general public. The majority of this diverse group of users has no physical impediments. They do not rely on any form of assistive technology to interact with EIT. It is for this group that applications have traditionally been developed.

With the emergence of new assistive technologies, the traditional user group has been expanded to include people with disabilities. Section 508 seeks to address this expansion of the user community by requiring applications be developed in a way that permits users of assistive technology access to these applications. It is important to note that Section 508 does not require software applications to be dull and unappealing. On the contrary, Section 508 promotes innovation by seeking to enhance software through increased functionality. The best software is ‘useable’ software.

### Who Was the “User” Before Section 508?

Until the passage of Section 508, Government IT as well as commercial IT was developed for the ‘perfect user’ or users who may be said to possess the following characteristics:

- Color recognition: ability to distinguish variation in hues which enables the reading of light blue text on a medium blue background
- Visual acuity: ability to quickly locate and identify items at a glance on an application page
- Tone and pitch recognition: identification of audible signals occurring within an application without difficulty regardless of tone or pitch
- Manual dexterity: ability to navigate and complete online forms before the page timed out

### Who Are the “Users” Now?

The majority of NASA HQ’s IT users have no physical limitations. However, they are not the only users. Advancements in assistive technology have enlarged the group of potential users and made these old assumptions false. For example:

- As we age our eyes tend to lose their ability to focus and may require corrective lens to see properly; such users may select large fonts to make text readable
- Some users with low vision may require the aid of a screen magnifier
- Some users may see with clarity but not be able to recognize all colors; reds and greens appear as shades of blue
- Some users may be blind and rely on a screen ‘reader’ to read the contents of the display to them or use a Braille to convert the screen text into a tactile message

- Applications that rely solely on a mouse or pointing device for input are inaccessible to some users who do not have full use of their hands or eyes.

To overcome a particular disability, many users must face the added challenge of learning and using assistive and adaptive technology products before they can even begin to interact with software applications. However, these technologies cannot overcome inaccessible designs within the software applications themselves.

## TECHNICAL STANDARDS

Web-based applications at NASA HQ combine database support with web delivery. This form of application is subject to a combination of the Section 508 requirements:

- 1194.22 for Web-based Applications
- 1194.21 for Software Applications

Since the applications tested are neither web pages (static pages) nor client-server software applications in the traditional sense, the complete set of accessibility standards for each form of IT is not applicable.

Below is a summary of the combined standards as they apply to the software tested in the development of the techniques shown in Appendix A – Technical Standards Table and a brief statement of explanation for each. The list is divided into two groups, one affecting application function and the other affecting application appearance or design.

### Functional Requirements

- **Keyboard use:** This provision requires that all user interaction with the application be supported through the keyboard. This does not forbid the use of mouse input but require that all such inputs may be accomplished through the keyboard. Section 508 - Software Applications and Operating Systems 1194.21(a)
- **Accessibility features:** This provision requires that the application not over ride the system configuration choice made by the user. Section 508 - Software Applications and Operating Systems 1194.21(b)
- **Skip Links:** This provision requires that a means of bypassing redundant links or controls be made available for users of assistive technology. Section 508 - Web-based Intranet and Internet Information and Applications 1194.22 (o)
- **Input focus:** This provision requires that the user be made aware of changes in display content or location when such changes are not detectable by assistive technology. Section 508 - Software Applications and Operating Systems 1194.21(c)
- **Timed Response:** This provision requires that applications, which utilize some form of user time or session control, provide alternatives for users who may require more than average time. Section 508 - Web-based Intranet and Internet Information and Applications 1194.22(p)

- **Scripts:** This provision requires that where scripts are used to display content, the content must be available in a form that is accessible to assistive technology. Section 508 - Web-based Intranet and Internet Information and Applications 1194.22(l)
- **Bitmap Images:** This provision requires that where images are used to identify application controls or status the use should be consistent throughout the application. Section 508 - Software Applications and Operating Systems 1194.21(e)
- **Textural Information:** This provision requires that all application text output be made available to the operating system's display function. Section 508 - Software Applications and Operating Systems 1194.21(f)

## Appearance Requirements

- **Frames:** This provision requires that all frames be named in way that identifies their purpose. Section 508 - Web-based Intranet and Internet Information and Applications 1194.22(i)
- **User Interface Elements:** This provision requires that where images or other forms of graphical representation are used to convey information or status, that that information be conveyed texturally as well. Section 508 - Software Applications and Operating Systems 1194.21(d)
- **Color Coding:** These provisions require that color alone cannot be used to convey information. Section 508 - Software Applications and Operating Systems 1194.21(i) & Section 508 - Web-based Intranet and Internet Information and Applications 1194.22 (c)
- **Color and Contrast:** Where an application permits the user to adjust color and contrast, the range of choices should be broad enough to accommodate low vision users. Section 508 - Software Applications and Operating Systems 1194.21(j)
- **Animation, Multimedia presentations:** Animations, movies and other media formats are not accessible to some disabled users. This provision requires that the information conveyed by the 'media' be displayed in an alternative accessible format. Section 508 - Software Applications and Operating Systems 1194.21 (h), Section 508 - Web-based Intranet and Internet Information and Applications 1194.22 (b)
- **Flashing, Flickering & Blinking:** Do not use any script, image animation or other device to create a Flashing, Flickering or Blinking effect. Section 508 - Software Applications and Operating Systems 1194.21(k) & Section 508 - Web-based Intranet and Internet Information and Applications 1194.22(j)

- **Tables:** Where tables are used to organize information either for page format or data display, the tables must be used in a way that supports assistive technology. Section 508 - Web-based Intranet and Internet Information and Applications 1194.22 (g & h)
- **Electronic Forms:** Forms should be created with coding that supports assistive technology, data fields that are accessible to disabled users and logical design for non-visual users. Section 508 - Software Applications and Operating Systems 1194.21 (l) & Section 508 - Web-based Intranet and Internet Information and Applications 1194.22 (n)
- **Client Side Image Maps:** Where images are used to convey information, the text equivalent must be provided for vision-disabled users. Section 508 - Web-based Intranet and Internet Information and Applications 1194.22(a) & 1194.22(f)

## DEVELOPMENT LIFECYCLE

For optimum integration, the accessibility requirements in Section 508 should be considered during all phases of the development lifecycle. This consideration will take different forms at different phases of development.

### Project Definition

If the project is conceived with user accessibility in mind, the overall accessibility of the application and compliance with Section 508 will be more successful and cost effective. When discussing a potential software development project with the customer (requester), attention should be given to the requirements of Section 508 as well as the application's intended user group. The fundamental requirement of Section 508 is to make all Government EIT accessible to disabled users. If the customer does not have specific knowledge of disabled users or Section 508's requirements they may be unaware of the need for compliance or the level of compliance required.

The requirements of Section 508 will impact the application's design, resource management and functional requirements. For example, a disabled user may require extended time to complete application interaction. This single requirement will impact:

- Security requirements. Longer session times mean sensitive information may stay on a user's workstation longer than current security limits permit.
- Resource requirements. Longer session times may require increased server support.
- Functional requirements: To adequately control session timing for a diverse group of users, the control of session time management may become an application requirement as well as a server security issue.

### Requirements Analysis

It is important to make the customer aware of Section 508 Requirements as they apply to the specific service request as well as the overall application's requirements. Awareness of Section 508 requirements at the outset will reduce design time and increase customer satisfaction with the completed application. The application of Section 508 requirements begins with the software requirements analysis process. Just as it is critical to have a clear understanding of the customer's requirements and expectations, it is imperative to have a clear understanding of Section 508's requirements. It is important to identify for the customer and the developer how the delivered application will meet the design and functional requirements of Section 508.

The system review for web-based applications should include browser selection that will support the accessible coding and design efforts of the developer. As stated previously, Internet Explorer (IE) has proven to be the best browser for support of assistive and adaptive technologies.

## Renovating Existing Web-Based Applications

The renovation of existing applications to comply with Section 508 Requirements will require careful scrutiny of the existing code and design. Application requirements before Section 508 relied heavily on the users ability to 'see' and use the application in the conventional sense. Large amounts of information could be displayed with confidence that the intended user could quickly and easily scan complex documents visually and derive meaning. The location of control items such as buttons, menus and animated graphics was located with attention to appearance rather than function and accessibility. Applications could utilize JavaScript enhancements that rapidly but inaccessibly manipulated large amounts of data and page content.

Most of these violations of Section 508 will be apparent. The renovation of these instances may be more problematic. Given the interdependence of application source code, isolating the necessary code, renovating it and testing its accessibility as well as functionality with dependent modules becomes a more difficult challenge. Using the Application testing procedure listed below along with the Accessibility Checklist and the Application Guide for the Technical Standards should provide the guidance to identify and make the necessary changes.

## Building Accessible Web-Based Applications

Appendix B -- Section 508 Compliance Checklist for Web-based Applications provides a list of criteria against which developers can evaluate their design and programming solutions. A clear understanding of the Section 508 requirements will enable a developer to design and code for user accessibility rather than around the accessibility requirements. Application interfaces and data display designs should reflect the concerns of Section 508:

- user interaction must be available for keyboard only users
- design color choices should be of sufficient contrast to enable low vision users access to information
- program output if in tables should be understandable if read linearly
- control items (buttons, menus, etc) should be consistently used through out the application

- when images, symbols, colors or other visual devices are used to convey information, the information must be available textually either through accessible code attributes or in screen text for users of assistive technology
- script enabled functions or page content changes must be apparent and predictable to disabled users

Application pages that meet the accessible display and functional requirements of Section 508 in the build phase will ideally comply with the Section 508 testing in the development test phase. Adhering to the Section 508 requirements in the coding phase of development will reduce the possibility of having to recode inaccessible pages detected during the development testing phase, thereby avoiding adverse project cost and scheduling impacts.

## Web-Based Application Testing

Prior to customer testing, the application should be tested and evaluated by the developer to determine compliance with Section 508 requirements. The Section 508 Compliance Checklist for Web-based Applications shown in Appendix B should be used to evaluate each page of the application. Web-based application modules that fail the test criteria outlined in the checklist should be subject to remediation and retested to confirm compliance. The final checklist test results should be included with the software documentation and delivered to the customer at the completion of the project. All testing should be performed with the baseline browser with the accessibility features enabled as shown below. The mouse should not be used during accessibility testing.

### **Enabling Accessibility Features in IE 5.5**

*To Enable ALT tag support (see Illustration A):*

- From the browser title bar, select "Tools"
- Select "Internet Options" tab
- Select "Always expand ALT text for images" to place a checkmark in the box
- Click OK

*To Enable Support for User Colors (see Illustration B):*

- Click on the tab marked "Colors"
- Click on each of the 3 boxes to ignore web page color, font size, and font style

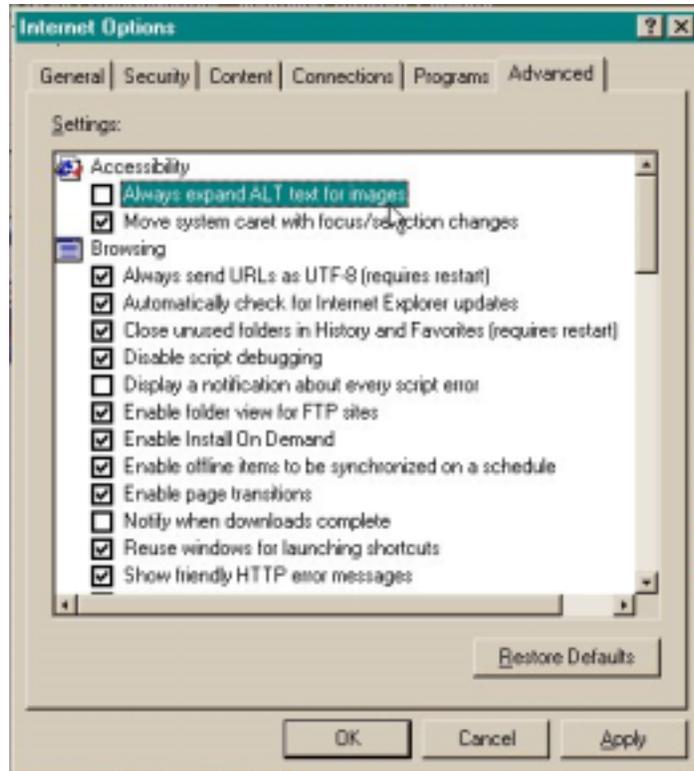


Illustration A

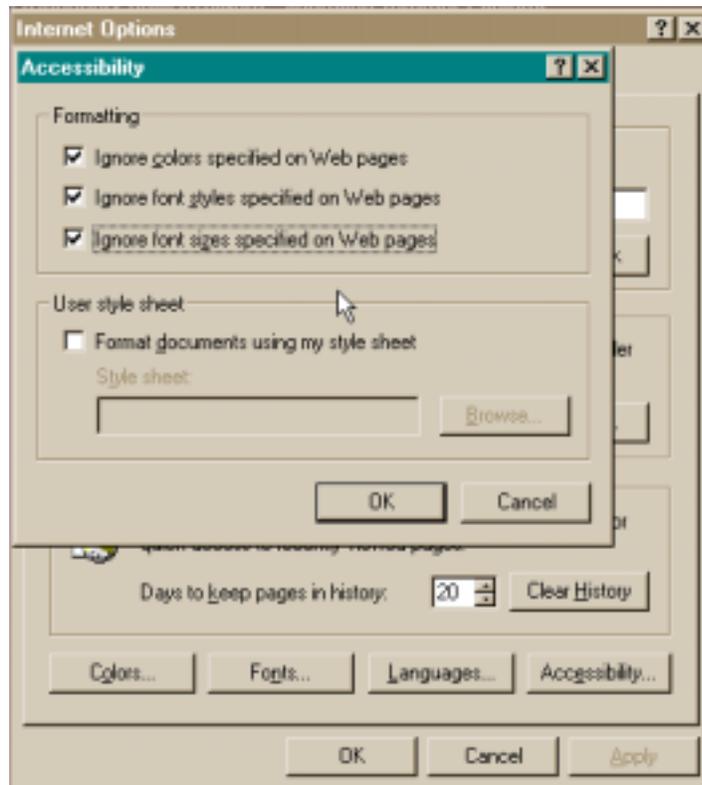


Illustration B

While the goal of accessible coding requirements may be clear, the interaction of new technology solutions with existing assistive technology may be unpredictable. It is recommended that some form of mainstream assistive technology product, such as WindowEyes by GW Micro, be used to test and establish the level of compliance these new solutions achieve. Where uncertainty exists as to the effectiveness of the developer's coding techniques, testing with the assistive technology ideally should be performed by a disabled user to gain an objective, third-party evaluation.

## SUMMARY

Compliance with the Section 508 Standards for Accessible Electronic Information Technology is mandatory for all Government IT procurements after June 21, 2001. The integration of Section 508 requirements at the start as well as throughout the software development lifecycle is necessary for the successful development of accessible software products.

The techniques and methods outlined in “Appendix A – Technical Standards” along with “Appendix B – Section 508 Compliance Checklist for Web-Based Applications” will serve to ensure compliance with Section 508 standards. The experience gained from the renovation of existing NASA HQ web-based applications will provide valuable lessons learned and improved methodology. These benefits will enable developers to create innovative and accessible solutions for NASA HQ’s IT needs.

APPENDIX A – TECHNICAL STANDARDS

Functional Requirements

<b>Standard</b>	Executing Function from Keyboard
<b>Regulation</b>	1194.21 (a) Software Applications and Operating Systems
<b>Rule</b>	When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.
<b>Explanation</b>	<p><b>Why is keyboard access to software required?</b></p> <p>When keyboard access to a program’s controls and features is provided, a person who cannot use a mouse or other pointing device will still be able to run the product. For example, a person with a disability that affects dexterity may find it impossible to move or hold a pointing device with enough accuracy to activate desired features. A person, who cannot see the screen, therefore relying on assistive technology, may have no problems moving the pointer but will be unable to determine where or to what the pointer is pointing.</p> <p><b>Does this provision prohibit the use of "mouse-only" functions in any software?</b></p> <p>All actions that can be identified or labeled with text are required to be executable from a keyboard. For example, most of the menu functions even in common drawing programs that allow a user to open, save, size, rotate, and perform other actions on a graphic image can all be performed from the keyboard. However, providing keyboard alternatives for creating an image by selecting a "drawing tool", picking a color, and actually drawing a design would be extremely difficult. Such procedures require the precise level of control afforded by a pointing device (e.g., a mouse) and cannot be given text labels because there is no way to predict what action the user plans to perform. Therefore, when a programmer is determining which functions need keyboard access, the best rule of thumb is to add keyboard shortcuts to any feature where the function can be identified with a text label.</p>
<b>Code Example</b>	<p>In this example, the page content is only available through the mouse. A keyboard only user would not be able to access the layered content that is triggered by mouse actions.</p> <pre>&lt;TD valign="top"&gt;&lt;a href="http://www.nasa.gov"&gt; &lt;img src="images/nasa_logo.jpg" width=59 height=48 border=0 alt="NASA Logo, links to NASA website"&gt;&lt;/A&gt;&lt;a href="http://www.earth.nasa.gov/Introduction/welcome.html"</pre>

	<p>OnMouseOver="document.welcome.src='images/welcome2.jpg';hide_all(); MM_showHideLayers('menuDiv1','show')"                  OnMouseOut="document.welcome.src='images/welcome.jpg';"&gt;</p>
<p><b>Screen Capture</b></p>	 <p>The screenshot shows the 'Destination: Earth' website. A mouse cursor is hovering over a link that says 'Image links to Earth Observatory'. Other visible text includes 'NASA NEWS RELEASE', 'earth breaking news from the earth observatory', 'HEALTHYPLANET', 'FOR KIDS ONLY kids.earth.nasa.gov', and '40 YEARS OF Earth Science LOOKING AT EARTH FROM SPACE'.</p>
<p><b>Renovated Code</b></p>	<pre>&lt;TD valign="top"&gt;&lt;a href="http://www.nasa.gov"&gt; &lt;img src="images/nasa_logo.jpg" width=59 height=48 border=0 alt="NASA Logo, links to NASA website"&gt;&lt;/A&gt;&lt;a href="http://www.earth.nasa.gov/Introduction/welcome.html" OnKeyDown="document.welcome.src='images/welcome2.jpg';hide_all(); MM_showHideLayers('menuDiv1','show')" OnKeyUp="document.welcome.src='images/welcome.jpg';"&gt;</pre>
<p><b>Screen Capture</b></p>	 <p>This screenshot is identical to the one above, but the mouse cursor is replaced by a white rectangular highlight, indicating that the link is now accessible via keyboard input.</p>
<p><b>Explanation</b></p>	<p>The page content is now available to keyboard input rather than mouse.</p>
<p><b>Code Identification</b></p>	<p>&lt;script&gt; OnMouse functions</p>

<p><b>Developer Suggestions</b></p>	<p>Barring obvious violations of this rule (mouse only accessibility), web-based applications may have pages with form elements within any number of frames. The tab key behavior is inherent to the browser and tabbing between frames on a page does not pose an accessibility limitation. The tab key simply selects the frame as a separate object and if there are form elements within the frame, tab will select those in tab order.</p> <p>However, the practicality of a page may become an issue. For example, a page contains form elements that require input and an action button (like “save”), which needs to be accessed for processing. Without the use of a mouse the user would have to traverse each field between the current form element to the save button. By using the “accesskey” attribute throughout the application, the user would simply (and consistently) use that feature rather than traversing frames and form elements to reach it.</p> <pre>&lt;a name="sr_save_link" href="whatever" accesskey="S"&gt;xyz&lt;/a&gt;</pre> <pre>&lt;cfset request.tab_counter=request.tab_counter+1&gt;</pre> <pre>&lt;input tabindex="#request.tab_counter#" type="text" name="#request.frm_elem#" size="15" value=""&gt;</pre> <p>The use of the “tabindex” attribute should also be used to specify a repeatable and logical tabbing order to all fields on a page to prevent confusion/frustration of a keyboard-only user. For dynamically created pages, a simple counter variable can be used to create the tab order in ascending order, for example:</p> <pre>&lt;a href="#" OnClick="return false;"&gt;Header&lt;/a&gt;</pre> <p>Another helpful tag is the anchor. For pages which are logically sectioned, the use of &lt;a&gt; on a section heading would provide a keyboard user a convenient tab-stop. By using the form given in the coding example above, the &lt;a&gt; will be regarded by the browser as a tab-stop and it will not respond to clicking (or spacebar/enter).</p>
-------------------------------------	--

<b>Standard</b>	Accessibility Features
<b>Regulation</b>	1194.21 (b) <i>Software Applications and Operating Systems</i>
<b>Rule</b>	Applications shall not disrupt or disable activated features of other products that are identified as accessibility features, where those features are developed and documented according to industry standards. Applications also shall not disrupt or disable activated features of any operating system that are identified as accessibility features where the application programming interface for those accessibility features has been documented by the manufacturer of the operating system and is available to the product developer.
<b>Explanation</b>	<p>The Access Board's concern here is that whatever control the application exerts over the operating system, it must not disable the user's chosen accessibility features. For example where an application offers font size control as an accessibility feature of the application, this control must not over ride the operating system's font display control settings in the OS accessibility configuration.</p> <p><i>"Paragraph (b) prohibits applications from disrupting or disabling activated features of other products that are identified as accessibility features, where those features are developed and documented according to industry standards. Applications also shall not disrupt or disable activated features of any operating system that are identified as accessibility features where the application programming interface for those accessibility features has been documented by the manufacturer of the operating system and is available to the product developer. The application-programming interface refers to a standard way for programs to communicate with each other, including the operating system, and with input and output devices. For instance, the application-programming interface affects how programs have to display information on a monitor or receive keyboard input via the operating system.</i></p> <p><i>Many commercially available software applications and operating systems have features built-into the programs that are labeled as access features. These features can typically be turned on or off by a user. Examples of these features may include, reversing the color scheme (to assist people with low vision), showing a visual prompt when an error tone is sounded (to assist persons who are deaf or hard of hearing), or providing "sticky keys" that allow a user to press key combinations (such as control-C) sequentially rather than simultaneously (to assist persons with dexterity disabilities). This provision prohibits software programs from disabling these features when selected. (See §1194.23(b)(2) in the NPRM.)</i></p> <p><i>Comment. The proposed rule only specified that software not interfere with features that affect the usability for persons with disabilities. Commenters from industry noted that the provision in the NPRM did not provide any method of identifying what features are considered access features and</i></p>

	<p><i>further stated that this provision was not achievable. These commenters pointed out that it was impossible for a software producer to be aware of all of the features in all software packages that could be considered an access feature by persons with disabilities. Sun Microsystems recommended that this provision address access features that have been developed using standard programming techniques and that have been documented by the manufacturer.</i></p> <p><i>Response. This provision has been modified in the final rule to reference access features that have been developed and documented according to industry standards. No other changes have been made in the final rule. “ *</i>  <i>*Comments from the Federal Access Board’s Final Rule on the Electronic and Information Technology Accessibility Standards</i></p>
<b>Code Example</b>	There are no examples of ColdFusion applications that offer or over ride the OS accessibility features.
<b>Renovated Code</b>	None available
<b>Screen Capture</b>	None available
<b>Explanation</b>	See explanation above
<b>Code Identification</b>	None available
<b>Developer Suggestions</b>	<p>In general ColdFusion can only return results (usually HTML) to the browser, there is nothing that ColdFusion can do to interfere with accessible functions in the OS or browser.</p> <p>Specifically, when the ColdFusion application returns java/VB-script, HTML, CSS, etc, these all are rendered by and run within the browser and thus fall under the control of browser accessibility functionality. However, a java applet or ActiveX control may have its own display routines and thus may not respect browser or OS settings. Such code is out of the scope of this suggestion area, but should be taken into consideration during an application 508 review.</p>

<b>Standard</b>	Skip Links
<b>Regulation</b>	1194.22 (0) Web-based Intranet and Internet Information and Applications
<b>Rule</b>	A method shall be provided that permits users to skip repetitive navigation links.
<b>Explanation</b>	<p>Sighted web users have no trouble visually scanning a page to locate the main content. The size, color, and placement of text easily identify its purpose without the user reading a word. For the disabled user who can not see the entire page, this can be a significant obstacle. They may rely on a screen reader to read the page to them. A disabled user may have to listen to half a page of titles, navigation links, and banners before they get to the main content. Providing hyperlinks that bypass redundant information can improve the accessibility of a page to a sight-impaired user.</p> <p>Page format can be a serious problem for visually disabled people because of how screen readers and talking browsers ‘speak’ a web page. A web page that is structured with layout tables relies on the users ability to make visual connections between the data presented. Screen readers and talking browsers use a technique called <b>linearizing</b> to convert a web page to a sequence of words and lines. The assistive technology first converts images to their alternative text. Then it spreads out (linearizes) the tables a cell at a time, working from left to right across each row and top to bottom. If a cell of a table contains a table, that table must be linearized before moving on to the next cell.</p> <p>On simple sites with layout tables, the screen is usually divided into a header that is a separate table (or perhaps not a table at all) and a table below that has a navigation panel in the cell on the left, and a wider content section in the cell on the right.</p> <p>Apply linearization to the simple site, and you see that first the header including its navigation gets spoken, then the navigation panel, then, at long last, the main content. If you are blind, chances are very good that you will have to listen to all the less important things before hearing what you want to hear.</p>
<b>Code Example</b>	<pre>&lt;table width="451" border="1" cellspacing="1" cellpadding="1" summary="This a table for page lay out purposes only"&gt; &lt;tr&gt; &lt;td colspan="3"&gt; &lt;div align="center"&gt;This is a title cell&lt;/div&gt; &lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td rowspan="2" width="123"&gt;Navigation link 1&lt;br&gt; Navigation link 2&lt;br&gt; Navigation link 3&lt;br&gt;</pre>

	<pre> Navigation link 4&lt;/td&gt; &lt;td height="65" width="224"&gt;Header Graphic&lt;br&gt; &lt;/td&gt; &lt;td rowspan="2" width="86"&gt;Right Nav 1&lt;br&gt; Right Nav 2&lt;br&gt; Right Nav 3&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td width="224"&gt;Main content area with lots of text and stories filling the center part of the window&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt; </pre>													
<p><b>Screen Capture (Non-Accessible)</b></p>	<table border="1" data-bbox="488 646 1247 835"> <tr> <td colspan="3" style="text-align: center;">This is a title cell</td> </tr> <tr> <td>Navigation link 1</td> <td rowspan="2" style="text-align: center;">Header Graphic</td> <td>Right Nav 1</td> </tr> <tr> <td>Navigation link 2</td> <td>Right Nav 2</td> </tr> <tr> <td>Navigation link 3</td> <td rowspan="2" style="text-align: center;">Main content area with lots of text and stories filling the center part of the window</td> <td>Right Nav 3</td> </tr> <tr> <td>Navigation link 4</td> <td></td> </tr> </table> <p>Linearization produces the following text (from IBM Home Page Reader):</p> <p>This is a title cell  Navigation link 1  Navigation link 2  Navigation link 3  Navigation link 4  Header Graphic  Right Nav 1  Right Nav 2  Right Nav 3  Main content area with lots of text and stories filling the center part of the window</p> <p>Note that the last to be spoken is the main content! We are looking for ways that users who are blind can skip all that navigation and advertising.</p>	This is a title cell			Navigation link 1	Header Graphic	Right Nav 1	Navigation link 2	Right Nav 2	Navigation link 3	Main content area with lots of text and stories filling the center part of the window	Right Nav 3	Navigation link 4	
This is a title cell														
Navigation link 1	Header Graphic	Right Nav 1												
Navigation link 2		Right Nav 2												
Navigation link 3	Main content area with lots of text and stories filling the center part of the window	Right Nav 3												
Navigation link 4														
<p><b>Renovated Code</b></p>	<p>There are 2 ways to make this table accessible.</p> <p>We can add an invisible 1px gif at the top of the page before the table with a link to an anchor at the beginning of the 'main content'.</p>													

 <a href="#">Skip to main content</a>		
This is a title cell		
Navigation link 1 Navigation link 2 Navigation link 3 Navigation link 4	Header Graphic  Main content area with lots of text and stories filling the center part of the window	Right Nav 1 Right Nav 2 Right Nav 3

We can also redesign the table in such a way that the main content cell will be encountered by the text reader before the text links.

(Read first) This is a title cell or Header Graphic	
(Read second) Main content area with lots of text and stories filling the center part of the window	(Read last) Right Nav 1 Right Nav 2 Right Nav 3  Navigation link 1 Navigation link 2 Navigation link 3 Navigation link 4

Or

(read first) This is a title cell or Header Graphic	
(read second)	(read last)
(read third)	Main content area with lots of text and stories filling the center part of the window
Navigation link 1 Navigation link 2 Navigation link 3 Navigation link 4	Right Nav 1 Right Nav 2 Right Nav 3

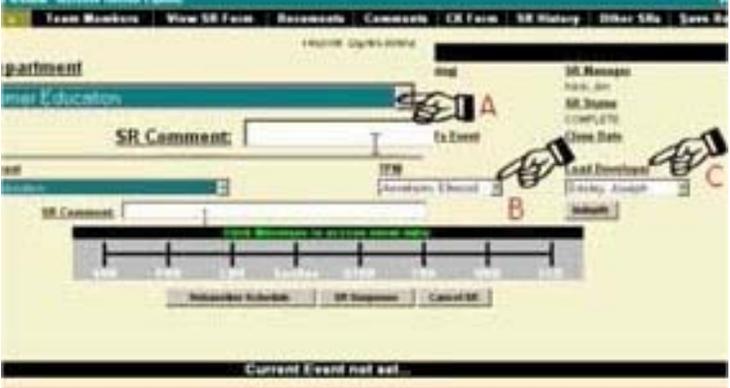
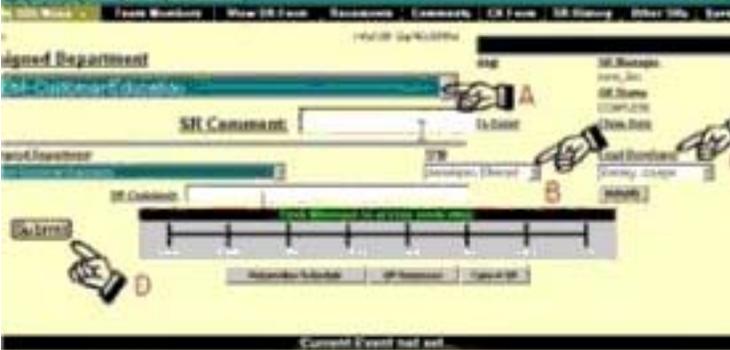
Another way of providing alternative navigation is with same color text and background. In the example below, a duplicate set of links are provided for screen reader users to jump over redundant information. Since the link text is the same color as the page background, the redundant links are not visible to sighted users and does not upset the look of the page.



To a screen reader user the top links are visible

	
<b>Explanation</b>	There are many ways to code navigation links so they do not adversely impact users of assistive technology. Test your page design by converting the table layout content to linear form and read it as a screen reader would to see if it makes sense.
<b>Code Identification</b>	Not applicable
<b>Developer Suggestions</b>	An easy way to preview the way a screen reader will render a page is to read the page's source code ignoring the HTML tags. The order the information is presented in the source is the same order the reader will follow.

<b>Standard</b>	Input Focus
<b>Regulation</b>	1194.21 (c) <i>Software Applications and Operating Systems</i>
<b>Rule</b>	A well defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that assistive technology can track focus and focus changes.
<b>Explanation</b>	<p>The concern of the Access Board is that by performing some keystroke or mouse action a change may occur to the page with out the user being aware of the change. In ColdFusion applications, most events are key or mouse driven and result in the page being reloaded. This automatic 'refreshing' of data informs the user that a 'change' has taken place. With proper page design and instructions, the user would be aware of the change and the nature of the change. JavaScript however is a client side scripting language and can change the application page's contents without re-loading the page. A screen-reading user would not be aware of the changes unless they were informed of this change and knew to reload the page in the screen reader's buffer.</p> <p>Also there are a number of client side event handlers that a visually disabled user would not be aware of. For example, the event handler OnChange when used to navigate between pages can be a problem. A keyboard only user could not move past the first item on a list enabled by the OnChange event handler. Likewise, this same event handler could be used to change the contents of a page with out reloading the page. Since the screen reader can only update page information if the page automatically 'reloads' or the user requests a 'reload' the user would be unaware of any page change unless the page explicitly indicated the change had taken place.</p> <p>When event handlers are used make sure that keyboard only users can navigate the pages as intended and that screen reader users and users of other assistive technologies are informed of content changes that may have only visual clues.</p> <p><i>"Paragraph (c) requires that software applications place on the screen a visual indication of where some action may occur if a mouse click or keystroke takes place. This point on a screen indicating where an action will take place is commonly referred to as the "focus". This provision also requires that the focus be readable by other software programs such as screen readers used by computer users who are blind. (See §1194.23(b)(3) in the NPRM.) No substantive comments were received and no changes have been made to this section in the final rule."</i></p>

<p><b>Code Example</b></p>	<pre>&lt;SELECT NAME="display_manager_id" SIZE=1 ONCHANGE="refresh_sr_list()"&gt;&lt;OPTION VALUE="458"&gt;Altner, Bruce&lt;OPTION VALUE="322"&gt;Anaheim, Ellwood&lt;OPTION VALUE="265"&gt;Arnold, Richard&lt;OPTION VALUE="500"&gt;Artemis, Diana&lt;OPTION</pre>
<p><b>Screen Capture</b></p>	 <p>Because of the OnChange event handler, this page's content changes in text boxes B and C as soon as the user attempts to tab through the drop down list in box A.</p>
<p><b>Renovated Code</b></p>	<p>The developer could change the event handler to OnKeyPress or add a submit button to overcome this accessibility barrier.</p>
<p><b>Screen Capture</b></p>	
<p><b>Explanation</b></p>	<p>By adding a submit button, the developer has returned page control to keyboard only users.</p>
<p><b>Code Identification</b></p>	<pre>&lt;script&gt; OnChange</pre>
<p><b>Developer Suggestions</b></p>	<p>If the app uses JavaScript to change page content, a java-alert (or similar) informing the 508 user that the page may need to be re-scanned by the screen reader should be provided. This alert should only be used in "508-mode" since such a message would confuse a user who is not using assistive technology (see Timed Response for discussion of 508 mode).</p> <p>It may also be necessary to issue such an alert if JavaScript is used to change the focus between multiple application (browser) windows.</p>

<b>Standard</b>	Timed Response
<b>Regulation</b>	1194.22 (p) <i>Web-based Intranet and Internet Information and Applications</i>
<b>Rule</b>	When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.
<b>Explanation</b>	<p>Accessibility problems can occur if a web page times-out while a user is completing a form. For security reasons or to reduce the demands on the server, you may be tempted to design pages with scripts so that they disappear or "expire" if a response is not received within a specified amount of time.</p> <p>A person with a disability may not be able to read, move around, or fill in a web form within the prescribed amount of time.</p> <p>The time available to read or complete an application page is generally controlled by the server on which the application is running. This time period is referred to as session timing. The session is regulated for a number of reasons.</p> <ol style="list-style-type: none"> <li>1. Security: If the application contains sensitive information, it would be unwise to allow information to remain on the display screen indefinitely where unauthorized users could gain access to it.</li> <li>2. Server Performance: A portion of server resources is allocated to each user's session. If sessions' timing were unregulated and left unclosed; the server's remaining available memory would run out of resources. Server performance would deteriorate and fail.</li> </ol> <p>These are both valid concerns. Section 508 however requires that allowances be made for users who may require more than average time to interact with an application. For example, a user who must use a pointer with a keyboard, or a screen reader may take longer to complete a page than is typical. For these users Section 508 requires the application to notify the user of an impending time out and give them the opportunity to extend their 'session' time to complete their work. One possible solution is to regulate session timing on the application level.</p> <p><i>Paragraph (p) addresses the accessibility problems that can occur if a web page times-out while a user is completing a form. Web pages can be designed with scripts so that the web page disappears or "expires" if a response is not received within a specified amount of time. Sometimes, this technique is used for security reasons or to reduce the demands on the computer serving the web pages. A disability can have a direct impact on the speed with which a person can read, move around, or fill in a web form. For this reason, when a timed response is required, the user shall be alerted and given sufficient time to indicate that additional time is necessary. (See §1194.21(d) in the NPRM.)</i></p> <p><i>Comment. The proposed rule prescribed specific settings for increasing the</i></p>

	<p><i>time-out limit based on a default setting. The Board sought comment on whether a system was commercially available that would allow a user to adjust the time-out. The Board also sought information on whether the proposed provision would compromise security. Commenters responded that security would be an issue if the time-out period was extended for too long and information with personal data was left exposed. Other commenters raised the point that specifying specific multiples of the default was unrealistic and arbitrary. The Multimedia Telecommunications Association (MMTA) stated that the default was not built-into a system. Rather, it was generally something that was set by an installer or a system administrator. They also noted that in order for a user to know that more time is needed, the user must be alerted that time is about to run out. (comments are from the Federal Access Board)</i></p>
--	--

<b>Developer Suggestions</b>	<p>Application developers/designers may wish to create a “508 mode”. This mode could be accessed by a visually handicapped user at an application entry point (i.e. the login page, home page, etc) or by a user profile if user login is used. The usual set of tags and attributes associated with 508 accessibility should be used throughout the application. Where there are certain pages (multi-frame, complex formatting tables, etc) that cannot support both visual and non-visual users, the 508 mode is “turned on”. Now the application will return a different set of pages that are 508 compliant thus leaving the visual pages intact. This may lead to 2 code bases, one being 508 compliant with the other being visually (i.e., non-508) designed. It is important to remember that Section 508 requires that both code bases be equally maintained (i.e., content updates made to one must also be made to the other). However, ColdFusion allows for modularity and code re-use. To incorporate such techniques a series of switches could be used to return the correct page type. In other words, the logic to retrieve and build the page is the same for each “version”, and a series of &lt;cfif&gt; (or other logic branching) statements would be used to control the returned 508/non-508 HTML.</p> <p>Another advantage to adding this type of mode would be for setting the application time-out. Since 508 calls for extended timeouts, the CFAPPLICATION tag could be set to an appropriately higher value than usual. Of course, the max timeout values on the ColdFusion server would have to be raised to allow for applications to achieve higher timeouts. Because of server resource and security politics, a “happy medium” point must be found between these timeout values and the ability of users to activate these higher timeouts. Users who do not have special needs should be programmatically discouraged from selecting a higher timeout simply for their convenience. For example, an anchor of a single pixel image with an &lt;alt&gt; attribute could be used as the trigger for higher timeout selection. A screen reader will read the &lt;alt&gt; as a matter of course, however a sighted user will only become aware of the switch with a mouse over &amp; tool-tip display, viewing the source HTML, or tabbing to it. For users with mobility difficulties, a button may be used to extend the session time limit.</p>
------------------------------	--

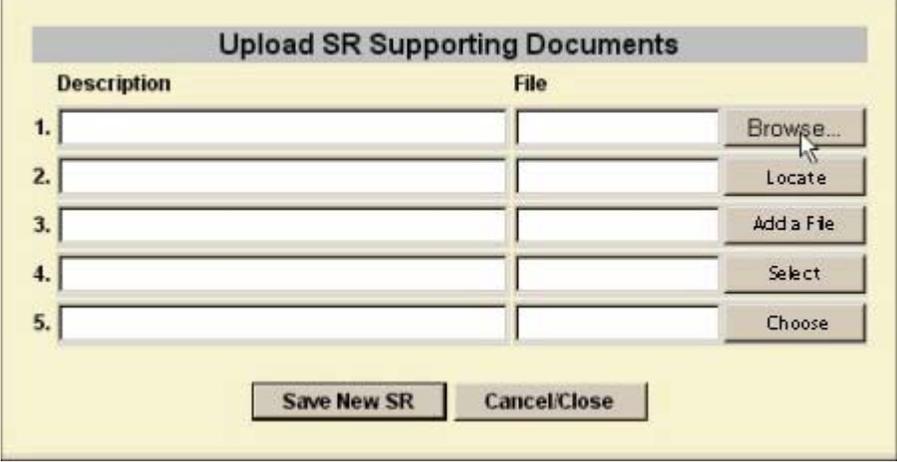
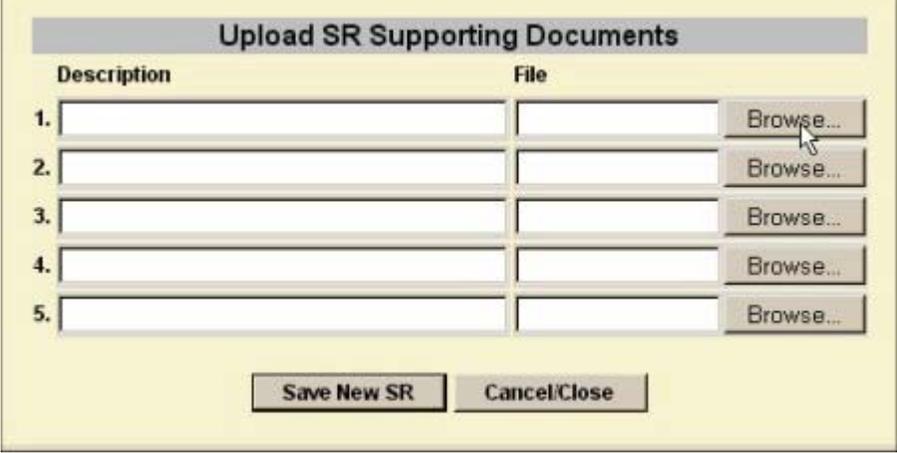
Standard	Scripts (see also Software Forms)
Regulation	1194.22 (l) <i>Web-based Intranet and Internet Information and Applications</i>
Rule	When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.
Explanation	<p><b>What accessibility problems can scripts cause?</b>          Web page authors have a responsibility to provide script information in a fashion that can be read by assistive technology. When authors do not put functional text with a script, a screen reader will often read the content of the script itself in a meaningless jumble of numbers and letters. Although this jumble is text, it cannot be interpreted or used.</p> <p><b>How can web developers comply with this provision?</b>          Web developers working with JavaScript frequently use so-called JavaScript URL's as an easy way to invoke JavaScript functions. Typically, this technique is used as part of &lt;a&gt; anchor links. For instance, the following link invokes a JavaScript function called myFunction:</p> <pre>&lt;a href="javascript:myFunction();"&gt;Start myFunction&lt;/a&gt;</pre> <p>This technique does not cause accessibility problems for assistive technology. A more difficult problem occurs when developers use images inside of JavaScript URL's without providing meaningful information about the image or the effect of the anchor link. For instance, the following link also invokes the JavaScript function myFunction, but requires the user to click on an image instead of the text "Start myFunction":</p> <pre>&lt;a href="javascript:myFunction();"&gt;&lt;img src="myFunction.gif"&gt;&lt;/a&gt;</pre> <p>This type of link, as written, presents tremendous accessibility problems, but those problems can easily be remedied. The &lt;img&gt; tag, of course, supports the "alt" attribute that can also be used to describe the image and the effect of clicking on the link. Thus, the following revision remedies the accessibility problems created in the previous example:</p> <pre>&lt;a href="javascript:myFunction();"&gt;&lt;img src="myFunction.gif" alt="picture link for starting myFunction"&gt;&lt;/a&gt;</pre> <p>Another technique advocated by some developers is to use the "title" attribute of the &lt;a&gt; tag. For instance, the following example</p>

	<p>includes a meaningful description in a "title" attribute:</p> <pre>&lt;a title="this link starts myFunction" href="javascript:myFunction();"&gt;&lt;img src="myFunction.gif"&gt;&lt;/a&gt;</pre> <p>This tag is supported by some but not all assistive technologies. Therefore, while it is part of the HTML 4.0 specifications, authors should use the &lt;alt&gt; tag in the enclosed image.</p> <p>Finally, the browser's status line (at the bottom of the screen) typically displays the URL of any links that the mouse is currently pointing towards. For instance, if clicking on an anchor link will send the user to <a href="http://www.usdoj.gov">http://www.usdoj.gov</a>, that URL will be displayed in the status line if the user's mouse lingers on top of the anchor link. In the case of JavaScript URL's, the status line can become filled with meaningless snips of script. To prevent this effect, some web developers use special "event handlers" such as OnMouseOver and OnMouseOut to overwrite the contents of the status line with a custom message. For instance, the following link will replace the content in the status line with a custom message "Nice Choice".</p> <pre>&lt;a href="javascript:myFcn();" OnMouseOver="status=Nice Choice'; return true;" OnMouseOut="status=";"&gt;&lt;img src="pix.gif"&gt;&lt;/a&gt;</pre> <p>This text rewritten into the status line is difficult or impossible to detect with a screen reader. Although rewriting the status line did not interfere with the accessibility or inaccessibility of the JavaScript URL, web developers should ensure that all important information conveyed in the status line also be provided through the "alt" attribute, as described above.</p> <p>JavaScript uses so-called "event handlers" as a trigger for certain actions or functions to occur. For instance, a web developer may embed a JavaScript function in a web page that automatically checks the content of a form for completeness or accuracy. An event handler associated with a "submit" button can be used to trigger the function before the form is actually submitted to the server for processing. The advantage for the government agency is that it saves government resources by not requiring the government's server to do the initial checking. The advantage for the computer user is that feedback about errors is almost instantaneous because the user is told about the error before the information is even submitted over the Internet.</p> <p>Web developers must exercise some caution when deciding which event handlers to use in their web pages, because different screen readers provide different degrees of support for different event handlers. The following table includes recommendations for using</p>
--	---

	<p>many of the more popular event handlers:</p> <p>OnClick – The OnClick event handler is triggered when the user clicks once on a particular item. It is commonly used on links and button elements and, used in connection with these elements, it works well with screen readers. If clicking on the element associated with the OnClick event handler triggers a function or performs some other action, developers should ensure that the context makes that fact clear to all users. Do not use the OnClick event handlers for form elements that include several options (e.g. select lists, radio buttons, checkboxes) unless absolutely necessary.</p> <p>OnDbClick – The OnDbClick event handler is set off when the user clicks twice rapidly on the same element. In addition to the accessibility problems it creates, it is very confusing to users and should be avoided.</p> <p>OnMouseDown and OnMouseUp – The OnMouseDown and OnMouseUp event handlers each handle the two halves of clicking a mouse while over an element – the process of (a) clicking down on the mouse button and (b) then releasing the mouse button. Like OnDbClick, this tag should be used sparingly, if at all, by web developers because it is quite confusing. In most cases, developers should opt for the OnClick event handler instead of OnMouseDown. The Section 508 keyboard standard requires all application functionality be made available through the keyboard as well as other input devices.</p> <p>OnMouseOver and OnMouseOut – These two event handlers are very popular on many websites. For instance, so-called rollover gif's, which swap images on a web page when the mouse passes over an image, typically use both of these event handlers. These event handlers neither can be accessed by the mouse nor interfere with accessibility – a screen reader simply bypasses them entirely. Accordingly, web designers who use these event handlers should be careful to duplicate the information (if any) provided by these event handlers through other means. The Section 508 keyboard standard requires all application functionality be made available through the keyboard as well as other input devices.</p> <p>OnLoad and OnUnload – Both of these event handlers are used frequently to perform certain functions when a web page has either completed loading or when it unloads. Because neither event handler is triggered by any user interaction with an element on the page, they do not present accessibility problems.</p> <p>OnChange – This event handler is very commonly used for triggering JavaScript functions based on a selection from within a</p>
--	--

	<p>&lt;select&gt; tag. Surprisingly, it presents tremendous accessibility problems for many commonly used screen readers and should be avoided. Instead, web developers should use the OnClick event handler (associated with a link or button that is adjacent to a &lt;select&gt; tag) to accomplish the same functions.</p> <p>OnBlur and OnFocus – These event handlers are not commonly used in web pages. While they don't necessarily present accessibility problems, their behavior is confusing enough to a web page visitor that they should be avoided.</p>
--	--

<b>Standard</b>	Bitmap Images
<b>Regulation</b>	1194.21 (e) Software Applications and Operating Systems
<b>Rule</b>	When bitmap images are used to identify controls, status indicators, or other programmatic elements, the meaning assigned to those images shall be consistent throughout an application's performance.
<b>Explanation</b>	<p><b>What forms of bitmap images are affected by this provision?</b>  This provision applies to those images that are used to indicate an action. An image used strictly for decoration is not covered by this provision.</p> <p><b>Why is the provision important for accessibility?</b>  Most screen reading programs allow users to assign text names to bitmap images. If the bitmap image changes meaning during a program's execution, the assigned identifier is no longer valid and is confusing to the user.</p> <p><i>Paragraph (e) requires that when bitmap images are used by a program to identify programmatic features, such as controls, the meaning of that image shall not change during the operation of a program. "Bitmap images" refer to a type of computer image commonly used in "icons" (e.g., a small picture of a printer to activate the print command). Most screen reading programs allow users to assign text names to bitmap images. If the bitmap image changes meaning during a program's execution, the assigned identifier is no longer valid and is confusing to the user. (See §1194.23(b)(6) in the NPRM.)</i></p> <p><i>Comment. As proposed, this provision did not identify which images had to remain consistent during the application. The AFB commented that the provision should be modified to indicate the type of image that needs to hold a consistent meaning during the running of an application. AFB noted that this provision should apply only to those bitmaps that represent a program function, and not to all images.</i></p> <p><i>Response. The final rule applies the provision to those images which are used to identify controls, status indicators, or other programmatic elements. No other changes have been made to this section in the final rule.</i></p>

<p><b>Screen Capture</b> <b>(Non-Accessible)</b></p>	<p>In the illustration below, the form control buttons all provide the same function. They open a selection menu from which the user may choose a file to upload. To a user who is unable to see the similarity of the form controls, this inconsistency of description may create confusion.</p> 
<p><b>Screen Capture</b> <b>(Accessible)</b></p>	
<p><b>Explanation</b></p>	<p>Be consistent with images and navigational elements. By using the same meaningful title for identical form controls the user is given a clear understanding of its function.</p> <p>Consistent use of images and navigational elements allow screen readers to provide additional navigational and identification aids to their users. Inconsistent use disrupts this flow, potentially confusing the user.</p>
<p><b>Code Identification</b></p>	<p>IMG Tags, form element identity elements</p>

<b>Developer Suggestions</b>	The consistent use of application control elements will benefit all users. This applies to the consistent naming of application controls as well as their placement. Where forms are used to display or collect data, consistent placement of form elements such as repetitive text boxes, submit and reset buttons will make the application more predictable and useable for disabled users.
------------------------------	--

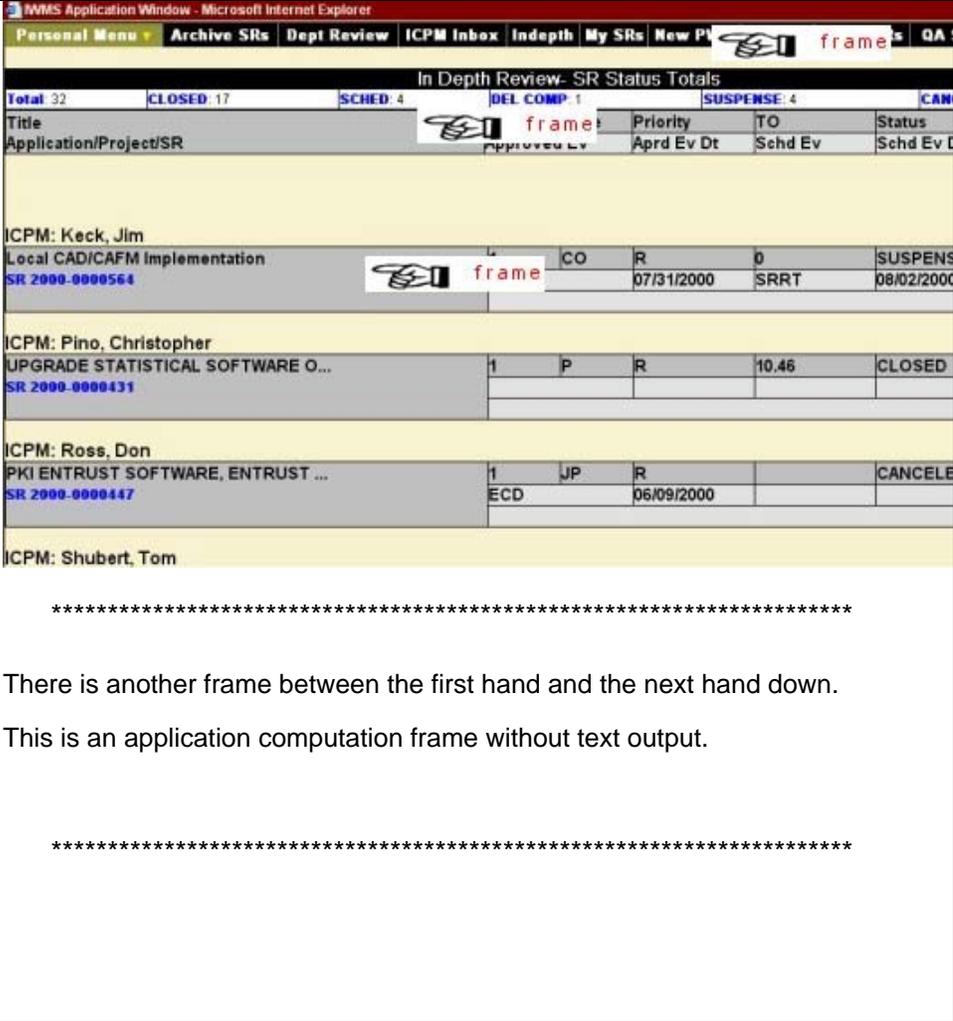
<b>Standard</b>	Textural Information
<b>Regulation</b>	1194.21 (f) <i>Software Applications and Operating Systems</i>
<b>Rule</b>	Textual information shall be provided through operating system functions for displaying text. The minimum information that shall be made available is text content, text input caret location, and text attributes.
<b>Explanation</b>	<p>This Standard requires that application output that is text, be sent to the computer screen regardless of what other form of output may be used. In this way, the application output will be accessible to users of assistive technology.</p> <p>Textual information must be displayed by the normal system operating routines. For example some encrypted forms of digital rights management (DRM) found typically in eBooks bypass the normal text display routines provided by the clients operating system to prevent unencrypted capture of their media. Example of this are encrypted forms of PDF and encrypted eBook reader found on handhelds such a Microsoft Reader.</p> <p>In addition, embedded forms of multimedia may not offer the full features of their standalone counterpart. For example, a video file may be provided with a text caption. If the file is referenced via an embedded link that caption may not be available since the availability of caption controls is controlled by the embedding code. Therefore, unless the caption is coded to be on by default, the embedded links may not be accessible.</p>
<b>Screen Capture (Non-Accessible)</b>	<p>The Adobe® Acrobat® eBook Reader™.</p>  <p>An example of an embedded QuickTime movie</p>

	
<p><b>Screen Capture (Accessible)</b></p>	<div data-bbox="516 674 911 1031">  </div> <p data-bbox="496 1073 1149 1100">Suggest download of Text-to-Speech plug-in for reader.</p> <div data-bbox="496 1157 878 1566">  </div> <p data-bbox="496 1608 1386 1675">Media with text caption should be turned on by default or downloaded to the stand-alone player.</p>
<p><b>Explanation</b></p>	<p data-bbox="496 1730 1321 1797">DRM media must provide their own 508 compliancy via their individual media player since the normal text OS functions may be bypassed.</p> <p data-bbox="496 1818 1344 1885">Media elements must be provided with either the ability to open communication with assistive technology or with that functionality 'on' by</p>

	<p>default in the case of embedded objects. Browser plug-in players may provide configuration options to enable support for 508 features.</p> <p>You must enable captioning in RealPlayer. In most versions of RealPlayer, you will find the enabling captions checkbox under the View/Preferences menu option. Then choose the Connection tab.</p>
<b>Developer Suggestions</b>	<p>Some web-based applications may have the ability for users to download or have sent files with MIME-types associated with non-browser applications (such as Acrobat Reader for PDF files, excel files, etc). These other applications should then be upgraded to 508 compliant versions.</p> <p>At present, three formats or languages support the ability to synchronize equivalent alternatives. These are Apple's QuickTime, the W3C's SMIL (Synchronized Multimedia Integration Language) and Microsoft's SAMI. For more information on captioning, see Standard: Multimedia.</p>

## Appearance Requirements

<b>Standard</b>	Frames
<b>Regulation</b>	1194.22 (i) <i>Web-based Intranet and Internet Information and Applications</i>
<b>Rule</b>	Frames shall be titled with text that facilitates frame identification and navigation.
<b>Explanation</b>	<p><b>Why is this provision necessary?</b></p> <p>Frames provide a means of visually dividing the computer screen into distinct areas that can be separately rewritten. Unfortunately, frames can also present difficulties for users with disabilities when those frames are not easily identifiable to assistive technology. For instance, a popular use of frames is to create "navigational bars" in a fixed position on the screen and have the content of the website retrievable by activating one of those navigational buttons. The new content is displayed in another area of the screen. Because the navigational bar doesn't change, it provides a stable "frame-of-reference" for users and makes navigation much easier. However, users with disabilities may become lost if the differences between the two frames are not clearly established.</p> <p><b>What is the best method for identifying frames?</b></p> <p>The most obvious way to accomplish this requirement is to include text within the body of each frame that clearly identifies the frame. For instance, in the case of the navigation bar, a web developer should consider putting words such as "Navigational Links" at the beginning of the contents of the frame to let all users know that the frame depicts navigational links. Providing titles like this at the top of the contents of each frame will satisfy these requirements. An additional measure that should be considered by agencies is to include meaningful text in the &lt;frame&gt; tag's "title" attribute. Although not currently supported by major manufacturers of assistive technology, the "title" attribute is part of the HTML 4.0 specification and was intended to let web developers include a description of the frame as a quote-enclosed string. Demonstrating the use of the "title" attribute requires a basic understanding of how frames are constructed. When frames are used in a web page, the first page that is loaded must include a &lt;frameset&gt; tag that encloses the basic layout of the frames on the page. Within the &lt;frameset&gt; tag, &lt;frame&gt; tags specify the name, initial contents, and appearance of each separate frame. Thus, the following example uses the "title" attribute to label one frame "Navigational Links Frame" and the second frame "Contents Frame."</p> <p>While assistive technology does not yet widely support the "title" attribute, we recommend including this attribute in web pages using frames.</p>

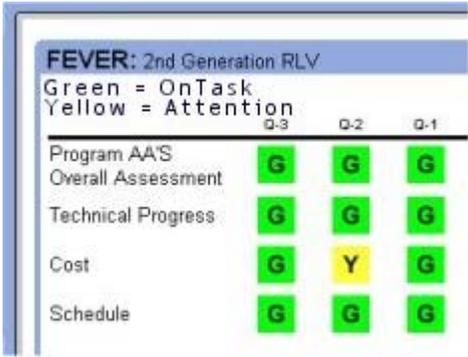
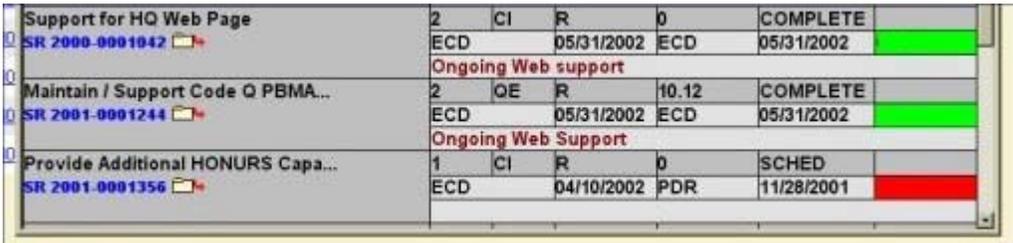
<p><b>Screen Capture</b></p>	 <p>*****</p> <p>There is another frame between the first hand and the next hand down. This is an application computation frame without text output.</p> <p>*****</p>
<p><b>Renovated Code</b></p>	<pre>&lt;frameset rows="40,0,*" border="0" frameborder="yes"&gt;     &lt;frame name="menubar" title="I W M S Main Menu"     src="iwms_menu.cfm?menu_type=#request.user_menu_type#"     marginheight="0" marginwidth="0" scrolling="#request.scrolling_type#"     noresize&gt;     ... &lt;/frameset&gt;</pre>
<p><b>Screen Capture</b></p>	<p>The title attribute for frames is not displayed visually in the browser window. It is recognized by screen reading software.</p>
<p><b>Code Identification</b></p>	<p>&lt;frame&gt;, &lt;frameset&gt;</p>

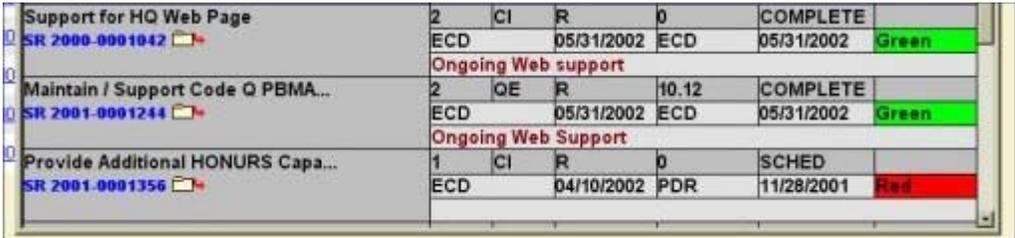
<b>Developer Suggestions</b>	<p>Using the 'title' attribute to describe frame purpose would be an easier 508 option instead of displaying text in the frame describing the frame purpose. Such text would change the current layout of the frame (for existing applications). Unfortunately, there is a potential visual consequence of using title in frames. A "tool-tip" may be displayed at the mouse pointer if the mouse is in a frame as the browser window is being drawn. Not a big deal in most instances, but some users who are unaware of Section 508 requirements may object.</p> <p>It would be a benefit to have standardized frame descriptions for consistency (i.e. "Application Menu Frame" for menus, "User Work Frame" for user input and work frames).</p> <p>Sometimes a hidden frame (frame width = 0) is used to send and receive application information/instructions from the server. This frame does not have any type of input or output (other than possibly JS). It is not displayed in the browser, but it is noticed by accessibility software like Window-Eyes. This software will simply read the frame when the page first loads, but will not reference it again since it contains no input or output elements. This practice as tested, does not violate any 508 guidelines.</p>
------------------------------	--

<b>Standard</b>	User Interface Element
<b>Regulation</b>	1194.21 (d) Software Applications and Operating Systems
<b>Rule</b>	Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.
<b>Explanation</b>	<p>Applications typically use buttons to indicate functions. A button may be an image file coded as a button or it may be graphical element created by the code itself, such as a Submit or Rest button. Regardless of the buttons source, the function of the button and its state must be communicated with text that can be accessed by assistive technology.</p> <p><i>'Paragraph (d) requires that software programs, through the use of program code, make information about the program's controls readable by assistive technology. Simply stated, this paragraph requires that information that can be delivered to or received from the user must be made available to assistive technology, such as screen reading software. Examples of controls would include button checkboxes, menus, and toolbars. For assistive technology to operate efficiently, it must have access to the information about a program's controls to be able to inform the user of the existence, location, and status of all controls. If an image is used to represent a program function, the information conveyed by the image must also be available in text. (See §1194.23(b)(4) and §1194.23(b)(5) in the NPRM.) No substantive comments were received and no changes have been made to this section, other than editorial changes.'</i></p>
<b>Code Example</b>	<p>In this example the 'back' button image of the browser uses a text message to indicate its status. The text message is the same when the button is enabled or disabled. This may confuse a user who is unable to 'see' the different images and must rely on the alternative text message for information.</p> <pre>&lt;img src="images/br_button_back_off.gif" alt="Go to previous page"&gt;</pre>

<p><b>Screen Capture</b> <b>(Non-Accessible)</b></p>	 <p>The screenshot shows the Netscape browser interface for 'NASA Headquarters'. The address bar contains 'http://www.hq.nasa.gov/'. A mouse cursor is hovering over a button labeled 'Go to previous page'.</p>
<p><b>Renovated Code</b></p>	<p>&lt;img src="images/br_button_back_off.gif" alt=" previous page unavailable"&gt;</p>
<p><b>Screen Capture</b> <b>(Accessible)</b></p>	 <p>The screenshot is identical to the one above, but the button text is now 'previous page unavailable'.</p>
<p><b>Explanation</b></p>	<p>By changing the alternative text to more accurately reflect the state of this control, users of assistive technology will understand the function of this application control.</p>
<p><b>Code Identification</b></p>	<p>Where images are used to indicate program controls, provide alt tag information to adequately indicate the control's state.</p>

<b>Developer Suggestions</b>	Possible examples: <ul style="list-style-type: none"><li>• Any images used in lieu of normal HTML objects (like a gif of a save icon instead of a save button) should have an alt attribute.</li><li>• Decorative images should be labeled (via alt) as such.</li><li>• Indicator images should be labeled via alt.</li><li>• Colors conveying meaning should have visible text associated with it.</li></ul>
------------------------------	---

<b>Standard</b>	Color (Coding)
<b>Regulation</b>	1194.21 (i) Software Applications and Operating Systems 1194.22 (c) Web-based Intranet and Internet Information and Applications
<b>Rule</b>	Color-coding shall not be used as the only means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.  Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.
<b>Explanation</b>	<p><b>How can color-coding create accessibility difficulties?</b></p>  <p>A software program that requires a user to distinguish between otherwise identical colored boxes or images for different functions (e.g., indicating status) would pose problems for anyone who was colorblind and would generally be very difficult to run with assistive technology. Screen reading software can announce color changes. However, this is an "on/off" feature. This means that if a user had to identify a specific color, they would have to have all colors announce which would greatly reduce the usability of the software for that person.</p> <p><b>Does the provision prohibit the use of colors?</b></p> <p>No. This provision does not prohibit the use of color to enhance identification of important features. It does, however, require that some other method of identification, such as text labels, be combined with the use of color.</p>
<b>Code Example</b>	<td headers="schedule_status" nowrap bgcolor="Lime">&nbsp; </td>
<b>Screen Capture (Non-Accessible)</b>	<p>This table uses color to identify the status of the data.</p> 
<b>Renovated Code</b>	<td headers="schedule_status" nowrap bgcolor="Lime">Green </td>

<p><b>Screen Capture</b> <b>(Accessible)</b></p>	<p>The renovated table includes text identification for the highlighted information and is now accessible and meaningful to users of assistive technology.</p> 
<p><b>Explanation</b></p>	<p>The use of color to visually indicate a change in status may be used if this change is conveyed texturally so that users who may not be able to distinguish between colors and users who rely on screen readers can perceive the change indicated.</p>
<p><b>Developer Suggestions</b></p>	<p>Where ever color is used to convey information or status of data that information or status must be communicated in a text manner as well.</p>

<b>Standard</b>	Color and Contrast Settings
<b>Regulation</b>	1194.21 (j) Software Applications and Operating Systems
<b>Rule</b>	When a product permits a user to adjust color and contrast settings, a variety of color selections capable of producing a range of contrast levels shall be provided.
<b>Explanation</b>	<p>This provision requires more than just providing color choices. The available choices must also allow for different levels of contrast. Many people experience a high degree of sensitivity to bright displays. People with this condition cannot focus on a bright screen for long because they will soon be unable to distinguish individual letters. An overly bright background causes a visual "white-out". To alleviate this problem, the user must be able to select a softer background and appropriate foreground colors. On the other hand, many people with low vision can work most efficiently when the screen is set with very sharp contrast settings. Because there is such a variance in individual needs it is necessary for a program to have a variety of color and contrast settings.</p> <p>Programs do have to offer a user a choice of interface colors; but if they do, the range of choices must be broad enough to accommodate users with low vision.</p> <p>Where background color and font color are controlled by an application, it is important to test the application for accessibility with assistive technology. Users who have difficulty with bright colors or with contrast may choose to disable the applications appearance settings by either browser configuration or the Operating System's display configuration. The application should be accessible with or with out the design color configuration.</p>
<b>Developer suggestions</b>	Certain color hues may not be perceptible to all people. By maintaining a varied contrast between different colors and hues, those differences may be emphasized. Converting your images, background colors and text color to grayscales will help identify color contrast problems.

<b>Standard</b>	Multimedia
<b>Regulation</b>	<i>1194.22 (b) Web-based Intranet and Internet Information and Applications</i>
<b>Rule</b>	Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.
<b>Explanation</b>	Captioning for the audio portion and audio description of visual information of multimedia presentations are considered equivalent alternatives. This provision requires that when an audio portion of a multimedia production is captioned, as required in provision (a), the captioning must be synchronized with the audio. Synchronized captioning would be required so someone reading the captions could also watch the speaker and associate relevant body language with the speech.

**Code Example**  
**(Non-Accessible)**

**Project: 01glidbg.mov**

**QuickTime, dur 03:13.093**      **time --:02:59.533**

Video: 320x180 pixels, 15.0031 fps, Sorenson Video

Display aspect ratio: raw pixels

Crop aspect ratio: Same as Display

Interlaced:

Audio: 16 bit mono, 22.050 kHz, QDM2, 352.8 kbits/s

3  
2  
1  
0 Mbps

▶ **Settings**      Edit

▶ **Settings Modifiers & Metadata**      Edit

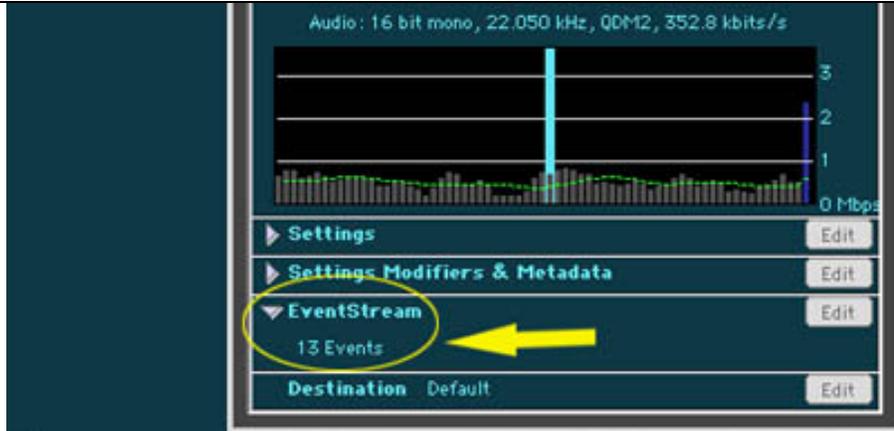
▼ **EventStream**      Edit

    No Events      ←

**Destination** Default      Edit



**Renovated Code**



The screenshot shows an audio player interface. At the top, it displays 'Audio: 16 bit mono, 22.050 kHz, QDM2, 352.8 kbits/s'. Below this is a waveform visualization. A settings menu is open, showing options for 'Settings', 'Settings Modifiers & Metadata', 'EventStream' (highlighted with a yellow circle and arrow), and 'Destination'. The 'EventStream' option indicates '13 Events'.

**EventStream**

Start	Event	Info
--:02:32.285	Display Text	When the Wrights tried to turn left, it would s
--:02:36.408	Display Text	Clearly there was something seriously wrong.
--:02:39.070	Display Text	The 1901 flying season ended with the Wright
--:02:42.402	Display Text	thoroughly confused and depressed.
--:02:46.280	Display Text	The 1901 Wright glider incorporated the sum
--:02:49.620	Display Text	of all 19th century aeronautical knowledge
--:02:52.322	Display Text	and still it wasn't enough to produce a machine
--:02:56.687	Display Text	It was a technological dead-end and the Wright
--:02:59.958	Display Text	In order to make further progress
--:03:01.828	Display Text	they would have to take a completely different
--:03:04.403	Display Text	and do some basic scientifi

Display Text ...

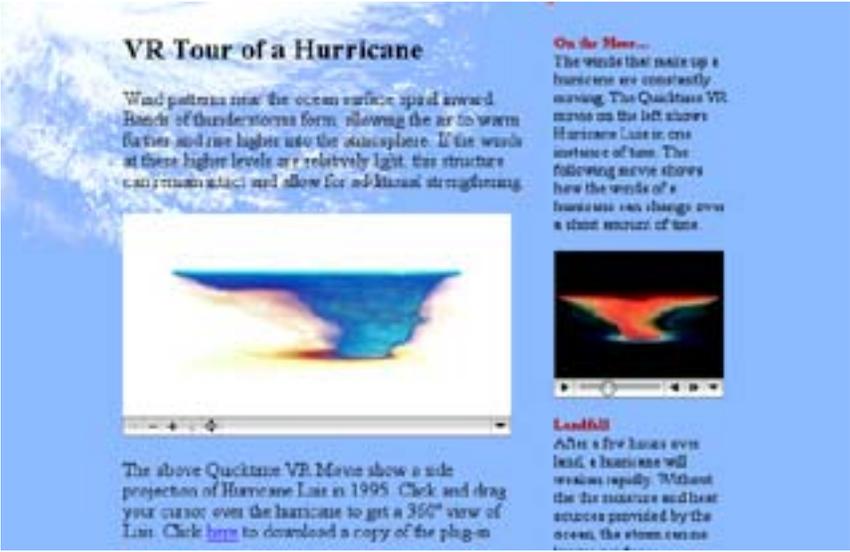
Start  Duration

Text

Add
Delete
Import...
Export...
Save

<p><b>Screen Capture (Accessible)</b></p>	
<p><b>Explanation</b></p>	<p>The first step in the process is to digitize the acquired video.</p> <p>The next step would be to transcribe the text of the video if the client does not provide a script. This may be done by either manually typing into a text editor or by listening to the movie and simultaneously speaking the text into a voice recognition program.</p> <p>If the action in the movie makes a significant contribution to the meaning of the clip, then a separate text description should be included and made available online.</p> <p>The final step of the text/video synching process would be to import the digitized video clip into Cleaner 5. From here you would choose to “Edit” the Event Stream. This is done by manually playing through the video and adding a “Display Text” event to correspond to the audio. Copy and paste the text from where you transcribed the audio.</p> <p>From Cleaner 5 you may encode to Quicktime, RealVideo, or Windows Media.</p>
<p><b>Code Identification</b></p>	<p>The way to identify whether a streaming video clip is Section 508 compliant is to visually look at the movie player and see if it has text captioning. There is no equivalent to “alt” tags embedded within the movie.</p>

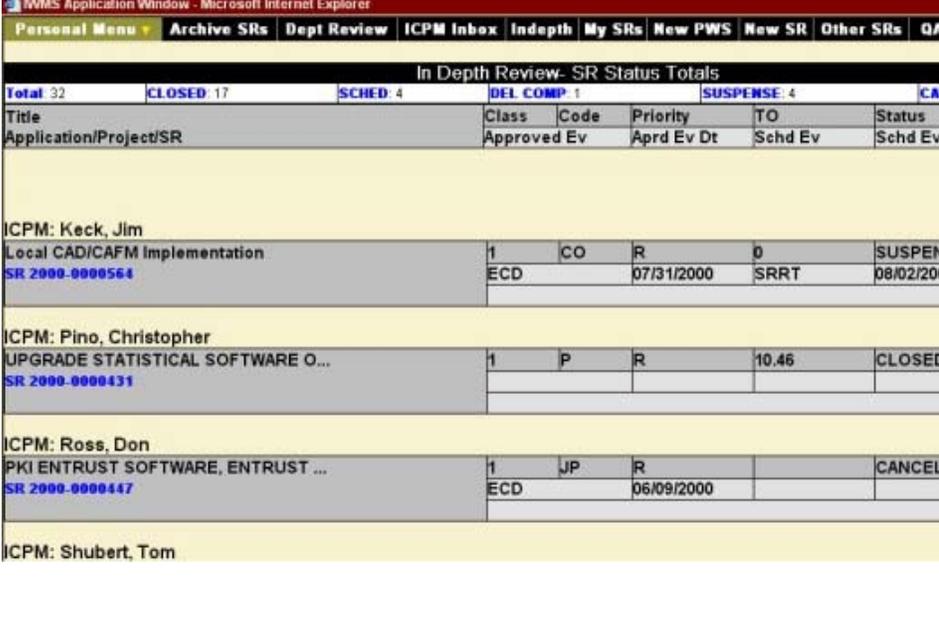
<b>Developer Suggestions</b>	<p>Spell check your document before synching it with the video.</p> <p>Use the standalone version of the player, not the plug-in (or embedded) version. This helps avoid problems with improperly configured plug-ins, and also maximizes the accessibility functions of the player.</p> <p>The amount of time it will take to transcribe and encode is roughly two and a half times as long as it would take to produce a clip without accessibility.</p> <p>When finished encoding, do a quality check to make sure the text matches the audio, that there are no words misspelled, and that ambient sounds such as background music are captioned as well.</p> <p>It is helpful to note when there is a change of speaker.</p>
------------------------------	---

<b>Standard</b>	Animation
<b>Regulation</b>	1194.21 (h) Software Applications and Operating Systems
<b>Rule</b>	When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.
<b>Explanation</b>	<p><b>Does this prohibit the use of animation?</b></p> <p>No, it simply requires that at least one non-animated presentation of the information conveyed by the animation be included.</p>
<b>Code Example</b>	<IMG SRC="animation.gif"> FLASH and Quicktime Movie
<b>Screen Capture (Non-Accessible)</b>	<p>Below is an inaccessible QuickTime movie demonstrating the creation of a hurricane.</p>  <p>The screenshot shows a web page with a blue background. At the top, it says 'VR Tour of a Hurricane'. Below this, there are two columns of text. The left column explains wind patterns near the ocean surface and how they form into a hurricane. The right column is titled 'On the Move...' and describes how the QuickTime VR movie shows the hurricane's path over time. In the center, there is a 3D visualization of a hurricane with a color gradient from blue to red. Below this visualization is a QuickTime player interface. At the bottom, there is a section titled 'Landfall' which explains that hurricanes weaken rapidly without moisture and heat from the ocean.</p>
<b>Renovated Code</b>	<A HREF="explain.html">Click here for an explanation of what you are seeing.</A>

<p><b>Screen Capture (Accessible)</b></p>	<p><b>A link has been added to a page offering the user an accessible alternative to the movie in the form of a text only description.</b></p>  <p>The screenshot shows a web page with a blue background. At the top, it says 'VR Tour of a Hurricane'. Below this, there is a paragraph of text describing the hurricane's structure. To the right, there is a smaller paragraph and a small image of a hurricane. At the bottom right, there is a button labeled 'Link to Description' with a hand cursor pointing to it. Below the button, there is a small note that says 'NOTE: After a few hours you find a hurricane will occasionally follow'.</p>
<p><b>Explanation</b></p>	<p>By including at least one non-animation presentation of the information conveyed by the animation. Persons using assistive technology can understand what information the animation is supposed to be conveying.</p>
<p><b>Code Identification</b></p>	<p>Animated images, FLASH, video</p>
<p><b>Developer Suggestions</b></p>	<p>Where applicable the use of the LONG DESC attribute may be appropriate or a text alternative page.</p>

<b>Standard</b>	Flicker or Blinking Text
<b>Regulation</b>	1194.21 (k) Software Applications and Operating Systems 1194.22 (j) Web-based Intranet and Internet Information and Applications
<b>Rule</b>	Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.
<b>Explanation</b>	<b>Why are flashing or blinking displays limited by this provision?</b> This requirement is necessary because some individuals with photosensitive epilepsy can have a seizure triggered by displays that flicker or flash, particularly if the flash has a high intensity and is within certain frequency ranges.
<b>Code Example</b>	<b>Non-accessible HTML example:</b> <pre>&lt;h1&gt;&lt;blink&gt;&lt;font color="#FF0000"&gt;Caution!!&lt;/font&gt;&lt;/blink&gt; &lt;/h1&gt;</pre> CSS example: <pre>&lt;style type="text/css"&gt; &lt;!-- blinktext { font-family: "Times New Roman", Times, serif; font-size: 24px; font-weight: bold; color: #FF6600; text-decoration: blink; background-color: #66FF00} --&gt; &lt;/style&gt; &lt;font face="Trebuchet MS, Arial, Verdana, MS Sans Serif" class="blink_text"&gt;Caution!!&lt;/font&gt;</pre>
<b>Screen Capture</b>	Not available
<b>Code Identification</b>	These blink attributes may be detected either: <ul style="list-style-type: none"> <li>• with a visual search with a Netscape browser</li> <li>• or, a text search of the source code for <pre>&lt;blink&gt;</pre> <pre>text-decoration: blink</pre>, check external style sheet files as well</li> </ul>

<b>Standard</b>	Tables
<b>Regulation</b>	1194.22 (g) & (h) <i>Web-based Intranet and Internet Information and Applications</i>
<b>Rule</b>	<p>Row and column headers shall be identified for data tables.</p> <p>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.</p>
<b>Explanation</b>	<p>Tables can be used for formatting purposes. Tables will place content on a page in a specific location. The accessibility issues regarding this use of tables are covered more fully in the standard covering <b>Skip Links</b>.</p> <p>Tables are also used to visually organize and present data on a page. The organizational relationship between rows and columns is apparent to a user who can see the table in its entirety. Where reference rows and or columns are present, the tables meaning is made clear quite quickly.</p> <p>A visually disabled user who uses a screen reader to 'read' the table is not afforded the benefit seeing the entire table and making the visual associations of information the table layout affords a sighted user.</p> <p>When a screen reader encounters a table, it will identify the number of rows and columns but the contents of the data cells will remain unknown until each cell is visited individually. The user will be given the option to identify a row, a column, both or neither as reference points for the table data. When this feature is enabled, the user will hear each cells contents read with its selected reference cell(s). This ability to associate data cells with reference points makes the table data accessible and meaningful.</p> <p>There are some table designs however that pose significant problems for users of screen readers. Tables that combine data in cells using &lt;colspan&gt; and &lt;rowspan&gt; can make it difficult for screen reader to render the complex informational relationships in a meaningful way. The absence of informative reference columns or rows decrease the accessibility of the table.</p> <p>Further it is recommended that developers include accessible HTML code attributes when creating data tables. The correct use of Scope and Header tag attributes will enable developing assistive technology to better render table data for disabled users.</p> <p><i>Paragraphs (g) and (h) permit the use of tables, but require that the tables be coded according to the rules of the markup language being used for creating tables. Large tables of data can be difficult to interpret if a person is using a non-visual means of accessing the web. Users of screen readers can easily get "lost" inside a table because it may be impossible to associate a particular cell that a screen reader is reading with the corresponding column headings and row names. For instance, assume that a salary table includes the salaries for federal employees by grade and step. Each row in the table may represent a grade scale and each column may represent a step. Thus, finding the salary corresponding to a grade 9, step 5 may involve finding the cell in the ninth row and the fifth column. For a salary chart of 15 grade scales and 10 steps, the</i></p>

	<p>table will have at least 150 cells. Without a method to associate the headings with each cell, it is easy to imagine the difficulty a user of assistive technology may encounter with the table.</p> <p>Section 1194.22 (g) and (h) state that when information is displayed in a table format, the information shall be laid out using appropriate table tags as opposed to using a preformatted table in association with the "&lt;pre&gt;" tag. Web authors are also required to use one of several methods to provide an association between a header and its related information.</p>																																																																																				
<p><b>Code Example</b></p>	<p>In this example, the developer has used ColdFusion to produce a series of tables. To reduce the amount of visibly redundant information, the developer has placed a descriptive header in one frame and the data tables in the lower scrolling frame. The table is further enhanced by splitting rows and combining columns allowing the user to display complex information in a very small area.</p>																																																																																				
<p><b>Screen Capture</b> <b>(Non-Accessible)</b></p>	 <p>The screenshot shows a web browser window titled 'IAMS Application Window - Microsoft Internet Explorer'. The navigation menu includes 'Personal Menu', 'Archive SRs', 'Dept Review', 'ICPM Inbox', 'Indepth', 'My SRs', 'New PWS', 'New SR', 'Other SRs', and 'QA'. The main content area is titled 'In Depth Review- SR Status Totals' and contains a table with the following data:</p> <table border="1"> <thead> <tr> <th>Total</th> <th>CLOSED</th> <th>SCHED</th> <th>DEL COMP</th> <th>SUSPENSE</th> <th>CAN</th> </tr> </thead> <tbody> <tr> <td>32</td> <td>17</td> <td>4</td> <td>1</td> <td>4</td> <td></td> </tr> </tbody> </table> <p>Below this is a table with columns: Title, Class, Code, Priority, TO, Status. The rows are:</p> <table border="1"> <thead> <tr> <th>Title</th> <th>Class</th> <th>Code</th> <th>Priority</th> <th>TO</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>Application/Project/SR</td> <td>Approved Ev</td> <td>Aprd Ev Dt</td> <td>Schd Ev</td> <td>Schd Ev D</td> <td></td> </tr> <tr> <td colspan="6">ICPM: Keck, Jim</td> </tr> <tr> <td>Local CAD/CAFM Implementation</td> <td>1</td> <td>CO</td> <td>R</td> <td>0</td> <td>SUSPENS</td> </tr> <tr> <td>SR 2000-0000564</td> <td>ECD</td> <td>07/31/2000</td> <td>SRRT</td> <td>08/02/2000</td> <td></td> </tr> <tr> <td colspan="6">ICPM: Pino, Christopher</td> </tr> <tr> <td>UPGRADE STATISTICAL SOFTWARE O...</td> <td>1</td> <td>P</td> <td>R</td> <td>10.46</td> <td>CLOSED</td> </tr> <tr> <td>SR 2000-0000431</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td colspan="6">ICPM: Ross, Don</td> </tr> <tr> <td>PKI ENTRUST SOFTWARE, ENTRUST ...</td> <td>1</td> <td>JP</td> <td>R</td> <td></td> <td>CANCELE</td> </tr> <tr> <td>SR 2000-0000447</td> <td>ECD</td> <td>06/09/2000</td> <td></td> <td></td> <td></td> </tr> <tr> <td colspan="6">ICPM: Shubert, Tom</td> </tr> </tbody> </table>	Total	CLOSED	SCHED	DEL COMP	SUSPENSE	CAN	32	17	4	1	4		Title	Class	Code	Priority	TO	Status	Application/Project/SR	Approved Ev	Aprd Ev Dt	Schd Ev	Schd Ev D		ICPM: Keck, Jim						Local CAD/CAFM Implementation	1	CO	R	0	SUSPENS	SR 2000-0000564	ECD	07/31/2000	SRRT	08/02/2000		ICPM: Pino, Christopher						UPGRADE STATISTICAL SOFTWARE O...	1	P	R	10.46	CLOSED	SR 2000-0000431						ICPM: Ross, Don						PKI ENTRUST SOFTWARE, ENTRUST ...	1	JP	R		CANCELE	SR 2000-0000447	ECD	06/09/2000				ICPM: Shubert, Tom					
Total	CLOSED	SCHED	DEL COMP	SUSPENSE	CAN																																																																																
32	17	4	1	4																																																																																	
Title	Class	Code	Priority	TO	Status																																																																																
Application/Project/SR	Approved Ev	Aprd Ev Dt	Schd Ev	Schd Ev D																																																																																	
ICPM: Keck, Jim																																																																																					
Local CAD/CAFM Implementation	1	CO	R	0	SUSPENS																																																																																
SR 2000-0000564	ECD	07/31/2000	SRRT	08/02/2000																																																																																	
ICPM: Pino, Christopher																																																																																					
UPGRADE STATISTICAL SOFTWARE O...	1	P	R	10.46	CLOSED																																																																																
SR 2000-0000431																																																																																					
ICPM: Ross, Don																																																																																					
PKI ENTRUST SOFTWARE, ENTRUST ...	1	JP	R		CANCELE																																																																																
SR 2000-0000447	ECD	06/09/2000																																																																																			
ICPM: Shubert, Tom																																																																																					
<p><b>Renovated Code</b></p>	<p>See "Developer Suggestions" below for a discussion of coding techniques. The following screen shot shows the "flattened" out frame. The header frame has been combined with the detail frame and the rowspan/colspan attributes have been removed. All other functionality is the same as the non-508 version.</p>																																																																																				

<p><b>Screen Capture</b></p>	
<p><b>Developer Suggestions</b></p>	<p>Not all adaptive technology utilize the tag attributes SCOPE and HEADER, but they are required by Section 508. What is important to remember is that a user who is visually disabled can not see the relationship between individual data cells and their reference column or header. Assistive technology can only relate data cells to their immediate the x and y reference points, or the row and or column bounding the table. Avoid the use of Split cells. Consider making a 'simple' table version of complex data that would be available for disabled users.</p> <p>Possible compliance: Since we are dealing with applications here, it may be beneficial to the application developer/designer to create a "508 mode". This mode could be activated by a non-sited user at an application entry point (i.e. the login page, home page, etc). The usual set of tags and attributes associated with 508 should be used throughout the application, however, there may be certain pages (multi-frame, complex formatting tables, etc) that cannot support both visual and non-visual aesthetics. In this case, when the 508 mode is "turned on", the application will return a different set of pages that are 508 compliant thus leaving the visual pages intact. This may lead to 2 code bases, one being 508 compliant with the other being visually (non-508) designed. However, ColdFusion allows for modularity and code re-use. Incorporating such techniques, a series of switches could be incorporated to return the correct page type. In other words, the logic to retrieve and build the page is the same for each "version", and a series of &lt;cfif&gt; (or other logic branching)</p>

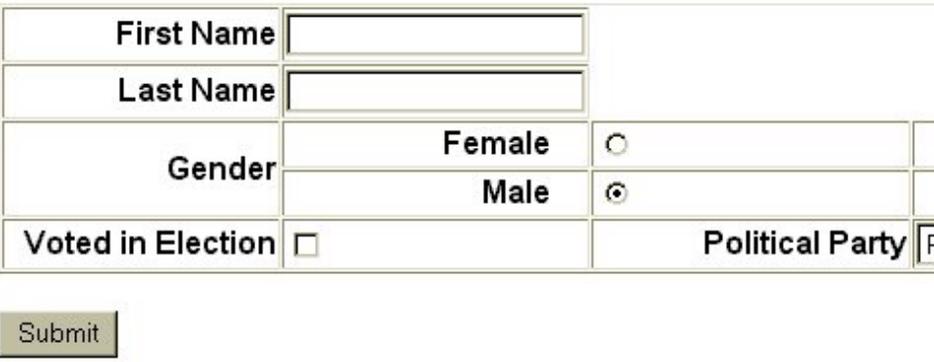
	<p>statements would be used to control the 508/non-508 HTML. Another advantage to adding this type of mode would be for setting the application timeout. Since 508 calls for extended timeouts, the CFAPPLICATION tag could be set appropriately.</p> <p>See Standard <b>Timed Response</b> for “508-mode” discussion.</p> <p>&lt;table summary=”explanation”&gt;</p> <p>One of the challenges disabled users face is identifying the intent of a data table from only the first few cells they ‘hear’ on either the ‘x’ or ‘y’ axis. For a sighted user the visual clues given by the row and column headings which can be ‘seen’ at a glance are often enough to gain a level of understanding about the table’s content. To assist disabled users there is the summary attribute for tables. This is a device similar to the ALT text description for images. It allows the programmer to provide some information to users of assistive technology that sighted users may not need. For developers who may have difficulty explaining the contents of a table, it may be advantageous to re-write the SQL Select statement that produced the table data. The identification of fields selected and the search criteria could supply the disabled user a short and effective description of the table’s purpose.</p>
--	---

<b>Standard</b>	Electronic Forms
<b>Regulation</b>	1194.22 (n) <i>Web-based Intranet and Internet Information and Applications</i> 1194.21 (l) <i>Software Applications and Operating Systems</i>
<b>Rule</b>	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.
<b>Explanation</b>	<p>Currently, the interaction between form controls and screen readers can be unpredictable, depending upon the design of the page containing these controls. HTML forms pose accessibility problems when web developers separate a form element from its associated label or title. For instance, if an input box is intended for receiving a user's last name, the web developer must be careful that the words "last name" (or some similar text) appear near that input box or are somehow associated with it. Although this may seem like an obvious requirement, it is extremely easy to violate because the visual proximity of a form element and its title offers no guarantee that a screen reader will associate the two or that this association will be obvious to a user of assistive technology.</p> <p>For example, where a table is used to locate form elements visually on a page, a disabled user would have difficulty associating the label of a text box if it were in an adjoining cell.</p> <p>For assured accessibility, it is recommended to add the Label attributes to all form elements. These attributes do not affect the appearance of the form but will enable users of assistive technology to access the form information with the least difficulty.</p>
<b>Code Example</b>	<p>The following form demonstrates these problems. Visually, this form is part of a table and each field is carefully placed in table cells adjacent to their corresponding labels (note: formatting forms with tables are by no means the only situation presenting the accessibility problems inherent in forms; tables merely illustrate the problem most clearly).</p> <p>While the relationship between the titles "First Name" or "Last Name" and their respective input boxes may be obvious from visual inspection, the relationship is not obvious to a screen reader. Instead, a screen reader may simply announce "input box" when encountering each input box. The reason for these difficulties is revealed from inspecting the HTML source for this table. The following code is a simplified version of this table.</p> <pre>&lt;form name="form1" method="post" action=""&gt;   &lt;table border="1" cellspacing="1" cellpadding="1"&gt;     &lt;tr&gt;       &lt;td width="138"&gt;</pre>

```

        <div align="right"><b>First Name</b></div>
    </td>
    <td width="135"> <b>
        <input type="text" name="textfield2">
    </b></td>
    <td width="159">
        <div align="right"><b>Last Name</b></div>
    </td>
    <td width="97"><b>
        <input type="text" name="textfield">
    </b></td>
</tr>
<tr>
    <td rowspan="2" bgcolor="#CCFFFF">
        <div align="right"><b>Gender</b></div>
    </td>
    <td width="135" bgcolor="#CCFFFF">
        <div align="right"><b>Female </b></div>
    </td>
    <td width="159" bgcolor="#CCFFFF"><b>
        <input type="radio" name="radiobutton" value="radiobutton">
    </b></td>
    <td width="97">&nbsp;  </td>
</tr>
<tr>
    <td width="135" bgcolor="#CCFFFF">
        <div align="right"><b>Male </b></div>
    </td>
    <td width="159" bgcolor="#CCFFFF"><b><b>
        <input type="radio" name="radiobutton" value="radiobutton">
    </b></b></td>
    <td width="97">&nbsp;  </td>
</tr>
<tr bgcolor="#FFFFCC">
    <td width="138" height="26">
        <div align="right"><b>Voted in Election</b></div>
    </td>
    <td width="135" height="26"> <b>
        <input type="checkbox" name="checkbox" value="checkbox">
    </b></td>
    <td width="159" height="26">
        <div align="right"><b>Political Party</b></div>
    </td>
    <td width="97" height="26"> <b>
        <select name="select">
            <option>undecided</option>
            <option>Republican</option>
            <option>Democrat</option>
            <option>Independant</option>
        </select>
    </b></td>
</tr>
</table>
<p>
    <input type="submit" name="Submit" value="Submit">
</p>

```

<p><b>Screen Capture</b> <b>(Non-Accessible)</b></p>	<p>&lt;/form&gt;</p> 
<p><b>Renovated Code</b></p>	<pre> &lt;form name="form1" method="post" action=""&gt; &lt;table border="1" cellspacing="1" cellpadding="1"&gt; &lt;tr&gt; &lt;td width="138"&gt; &lt;div align="right"&gt;&lt;label for="first"&gt;&lt;b&gt; First Name&lt;/b&gt;&lt;/label&gt;&lt;/div&gt; &lt;/td&gt; &lt;td width="135"&gt; &lt;b&gt; &lt;input type="text" name="textfield2" id="first"&gt; &lt;/b&gt;&lt;/td&gt; &lt;td width="159"&gt; &lt;div align="right"&gt;&lt;label for="last"&gt;&lt;b&gt;Last Name&lt;/b&gt;&lt;/label&gt;&lt;/div&gt; &lt;/td&gt; &lt;td width="97"&gt;&lt;b&gt; &lt;input type="text" name="textfield" id="last"&gt; &lt;/b&gt;&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td rowspan="2" bgcolor="#CCFFFF"&gt; &lt;div align="right"&gt;&lt;b&gt;Gender&lt;/b&gt;&lt;/div&gt; &lt;/td&gt; &lt;td width="135" bgcolor="#CCFFFF"&gt; &lt;div align="right"&gt;&lt;label for="female"&gt;&lt;b&gt;Female &lt;/b&gt;&lt;/label&gt;&lt;/div&gt; &lt;/td&gt; &lt;td width="159" bgcolor="#CCFFFF"&gt;&lt;b&gt; &lt;input type="radio" name="radiobutton" value="radiobutton" id="female"&gt; &lt;/b&gt;&lt;/td&gt; &lt;td width="97"&gt;&amp;nbsp;&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td width="135" bgcolor="#CCFFFF"&gt; &lt;div align="right"&gt;&lt;label for="male"&gt;&lt;b&gt;Male &lt;/b&gt;&lt;/label&gt;&lt;/div&gt; &lt;/td&gt; &lt;td width="159" bgcolor="#CCFFFF"&gt;&lt;b&gt;&lt;b&gt; &lt;input type="radio" name="radiobutton" value="radiobutton" id="male"&gt; &lt;/b&gt;&lt;/b&gt;&lt;/td&gt; &lt;td width="97"&gt;&amp;nbsp;&lt;/td&gt; &lt;/tr&gt; &lt;tr bgcolor="#FFFFCC"&gt; &lt;td width="138" height="26"&gt; &lt;div align="right"&gt;&lt;b&gt;Voted in Election&lt;/b&gt;&lt;/div&gt; &lt;/td&gt; &lt;td width="135" height="26"&gt; &lt;b&gt; &lt;label for="ck2"&gt;check if voted&lt;/label&gt;&lt;input type="checkbox" name="checkbox" value="checkbox" id="ck2"&gt; &lt;/b&gt;&lt;/td&gt; &lt;td width="159" height="26"&gt; &lt;div align="right"&gt;&lt;b&gt;Political Party&lt;/b&gt;&lt;/div&gt; &lt;/td&gt; &lt;td width="97" height="26"&gt; &lt;b&gt; &lt;select name="select"&gt; &lt;option&gt;Click here for undecided&lt;/option&gt; &lt;option&gt;Republican&lt;/option&gt; &lt;option&gt;Democrat&lt;/option&gt; &lt;option&gt;Independent&lt;/option&gt; &lt;/select&gt; &lt;/b&gt;&lt;/td&gt; &lt;/tr&gt; </pre>

	<pre> &lt;/table&gt; &lt;p&gt; &lt;input type="submit" name="Submit" value="Submit"&gt; &lt;/p&gt; &lt;/form&gt; </pre>
<p><b>Screen Capture</b></p>	<p>Form will appear the same in the browser window.</p>
<p><b>Developer Suggestions</b></p>	<p>Each form field must be visited to enter a unique label tag and the ID attribute must be added to each corresponding Form element to relate the text with the form field.</p> <p>This tag could be generated by ColdFusion for dynamic pages like the other HTML parts of the page would be, keeping in mind that a counter variable (or similar idea) should be used to keep each label name unique.</p> <p>A counter may also need to be used for ColdFusion dynamically generated pages to ensure tab index order. If the fields are not dynamic the tab order would be set with the other HTML &lt;input&gt; attributes.</p> <p><b>Example:</b></p> <p style="text-align: center;"><u>Before</u></p> <p>Phone Number: &lt;input type="text" name="phone_number" size="15"&gt;</p> <p style="text-align: center;"><u>After</u></p> <pre> &lt;cfset request.label_counter=request.label_counter+1&gt; &lt;cfset request.tab_counter=request.tab_counter+1&gt; &lt;cfoutput&gt; &lt;label name="label#request.label_counter#"&gt;Phone Number:&lt;/label&gt; &lt;input id="label#request.lavel_counter#" tabindex="#request.tab_counter#" type="text" name="phone_number" size="15"&gt; &lt;/cfoutput&gt; </pre>

<b>Standard</b>	Client-side Image Maps
<b>Regulation</b>	1194.22 (f) Web-based Intranet and Internet Information and Applications
<b>Rule</b>	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.
<b>Explanation</b>	<p>Client-side image maps allow a developer to assign text to each image map "hot spot." This feature means that someone using a screen reader can easily identify and activate regions of the map. An explanation of how these image maps are constructed will help clarify this issue.</p> <p>Creating a basic client-side image map requires several steps:</p> <ul style="list-style-type: none"> <li>• First, an image must be used in a client-side image map. This image is identified using the &lt;img&gt; tag. To identify it as a map, use the "usemap" attribute.</li> <li>• Use the &lt;MAP&gt; tag to "areas" within the map . The &lt;MAP&gt; tag is a container tag that includes various &lt;AREA&gt; tags that are used to identify specific portions of the image.</li> <li>• Use &lt;AREA&gt; tags to identify map regions . To identify regions within a map, simply use &lt;AREA&gt; tags within the &lt;MAP&gt; container tags. Making this client-side image map accessible is considerably easier to describe: simply include the "ALT" attribute and area description inside each &lt;AREA&gt; tag. The following HTML demonstrates how to make a client-side image map:</li> </ul>
<b>Code Example</b>	<pre>&lt;img src="navbar.gif" border="0" usemap="#Map"&gt; &lt;map name="Map"&gt; &lt;area shape="rect" coords="0,2,64,19" href="general.html" alt="information about us" &gt; &lt;area shape="rect" coords="65,2,166,20" href="jobs.html" alt="job opportunities" &gt; &lt;area shape="rect" coords="167,2,212,19" href="faq.html" alt="Frequently Asked Questions" &gt; &lt;area shape="rect" coords="214,2,318,21" href="location.html" alt="How to find us" &gt; &lt;area shape="rect" coords="319,2,399,23" href="contact.html" alt="How to contact us" &gt; &lt;/map&gt;</pre>

APPENDIX B – SECTION 508 COMPLIANCE CHECKLIST FOR WEB-BASED APPLICATIONS

CRITERIA	YES	NO	N/A
Are all functions and user input fields accessible to the keyboard? [Soft (a)]			
Are all program features available to the user when the user configured operating system and or browser accessibility features are enabled? (os:high contrast, font attribute selection and screen magnification & browser: user selected font type& size, colors and style sheet support) [Soft (b & g), Web (d)]			
Is the user made aware of changes to window content when page content changes as a result of active scripting rather than screen refreshing? [Soft (c), Web (l)]			
Are all application control elements represented by accessible text? [Soft (d)]			
Are all bitmap images representing application controls, indication of status or program elements used consistently through out the application? [Soft (e), Web (e & f)]			
Are all program output messages to the user in the form of displayed text or if an image is used to convey a message is the image accompanied by screen displayed text? [Soft (f)]			
If animation in the form of .gif, .avi, .mpg or other format is used to communicate information, is this information available in an accessible format such as .txt, .htm, .doc, etc? [Soft (h), Web (b)]			
When color is used to communicate information, is this information available in a different mode? [Soft (i), Web (c)]			
If the program permits the user to select interface colors does the range of colors available include selections for users with low vision? [Soft (j)]			
Does the application use prohibited flashing or blinking text, objects or elements? [Soft (k), Web (j)]			
Do all forms utilize the Label attribute, are form elements organized logically and consistently? [Soft (l), Web (n)]			
Do all images have ALT descriptive tags? [Web (a)]			
If a table is used for page format purposes, can it be read linearly (left to right and top to bottom) by a screen reader? Is the tables purpose identified with the summary attribute? [Web (g)]			
If a table is used to display data, does the table design and coding support interpretation by assistive technology? Are the reference row and column coded with the required identification code? [Web (h)]			
Are frames identified with the title attribute? Is the title descriptive of the frame's purpose? [Web (i)]			
Where time allowed for user interaction is controlled by the application (either client side or server side) is user given sufficient warning of time out status and is there a mechanism to extend session time and complete the task? [Web (p)]			

*\*Letter references shown in [ ] map to Section 508 Federal Access Board Technical Standards, i.e., Web (standard): Section 1194.22, Web-based Intranet and Internet Information; and, Applications and Soft (standard): Section 1194.21, Software Applications and Operating Systems*