Experiences in Accessibility and 508-Compliance Testing at RTI

Al-Nisa Berry, Emily Caron, Melissa Page, and Rhymney Weidner, RTI International

1. Abstract

As surveys become increasingly web-based, and the federal government—as well as other clients—frequently require accessibility standards to be met, it grows ever more important for developers to have flexibility and relative ease in applying accessibility-related attributes to survey controls. Blaise 5 includes many new features to help foster the development of an accessible survey; however, there remains room for improvement. This paper will cover some of the discrepancies between RTI's testing with ANDI (Accessible Name & Description Inspector) and what passed for the NVDA (NonVisual Desktop Access) screen reader. It will also describe the overall challenges we faced, along with some solutions applied while developing a 508-compliant web survey in Blaise 5.

2. Introduction

While we have used Blaise 5 successfully on many projects, in fall 2022, we endeavored to use it on our first project with the goal of being completely 508 compliant. Blaise 5.12.8 was the latest production version of Blaise 5 available when we began serious development, and so it was the version selected for use on this project. The survey in question was to be completed in CAWI and CATI mode. CAWI mode was designed with two sets of layout templates, one "large" for browsers with a width greater than 600 px and one "small" for browsers with a width less than 600 px. CATI mode would also be completed in a web browser, with slight adjustments to the layout and text compared to what was used for the CAWI "large" version. 508-compliance testing was conducted only on the CAWI "large" layout set and evaluated using the Chrome browser. 508 guidelines require the survey to follow all WCAG 2.0 rules to be compliant.

The 508-compliance testing team at RTI International, which is comprised of RTI QA staff, utilized the accessibility testing tool created by the Accessible Solutions Branch of the Social Security Administration called "ANDI." This tool is a Javascript-based tool that is easily installed in a variety of browsers, including Chrome, Edge, Firefox, Safari, and Internet Explorer.

While we encountered many issues during our journey to 508 compliance, this paper will cover representative issues from each of these areas: compliance issues that could not be solved, issues where we found workarounds to achieve compliance, and issues where we communicated with the Blaise 5 development team for potential solutions.

3. Testing Results

At every stage, the 508-compliance testing team provided testing results in easy-to-read Excel and Word documents. The Excel document listed each WCAG 2.0 requirement and the results for that item in the tested survey. Any issues discovered were listed in more detail in a separate Word document that provided a detailed description of the issue, along with screenshots. The screenshots provided specific screens for testing potential fixes, as well as making it clear what features in ANDI were used to identify the problem.

Figure 3a. An Example of Results Provided in Excel after Completing 508-Compliance Testing

Section 508 Conformance Test Process for Web Applicatic					
Section 508 Compliance Test Details Report, v1.0					
WCAG 2.0 Requirement No.	**WCAG 2.0 Test Name**	**Compliant? (Yes,No,N/A)**	**Test Date**	**Tester**	**Comments**
1.3.1	Programmatic Label	FAIL	12/8/2022	Aberry	Issue #8
3.2.2	On Input	PASS	12/8/2022	Aberry	

Figure 3b. An Example of a More Detailed Issue Description, Which Was Provided in a Separate Word Document; The Word Document Detailed Results Also Provide Screenshots to Help Isolate Fields/Screens Where the Problem Was Located

---

Issue #8:

WCAG 2.0 – 1.3.1 – Programmatic Label

Issue Description (Include screenshots): The programmatic labels for these fields seem inaccurate and may be confusing to respondents using assistive technology. The radio buttons are identified as 'invalid entry', and 'undefined of 0'. In addition, the label for each response states: "Invalid entry" (See examples below for details).

In addition, Improper use of [aria-labelledby] possible: Referenced ids ""Language, Selector"" not found error detected (See Ex. 1).

4. Initial Layout Testing

Before development began on the full instrument, we developed a small test instrument in Blaise 5.12.7 that contained one of each question type expected in the real survey. Early testing revealed issues in our test project. Some were issues that could be easily corrected. For example, error messages must be descriptive and specifically state what needs to be entered to correct the issue. This could be solved by implementing a role specifically for error text and making sure it was utilized where necessary. One caveat with this correction, however, is that additional role texts will require translation for surveys conducted in more than one language.

A few issues were quickly determined to be bugs and were corrected by Team Blaise in the next build of Blaise 5 (Blaise 5.12.8). Other issues were not so easy to solve, and some we were unable to resolve. Examples of these issues are demonstrated in the figures below.

Figure 4a. Programmatic Labels Were Not Accurate; For Example, Radio Buttons Were Labeled as "Undefined of 0" Instead of a Name That Described the Option That Would Be Selected by That Radio Button

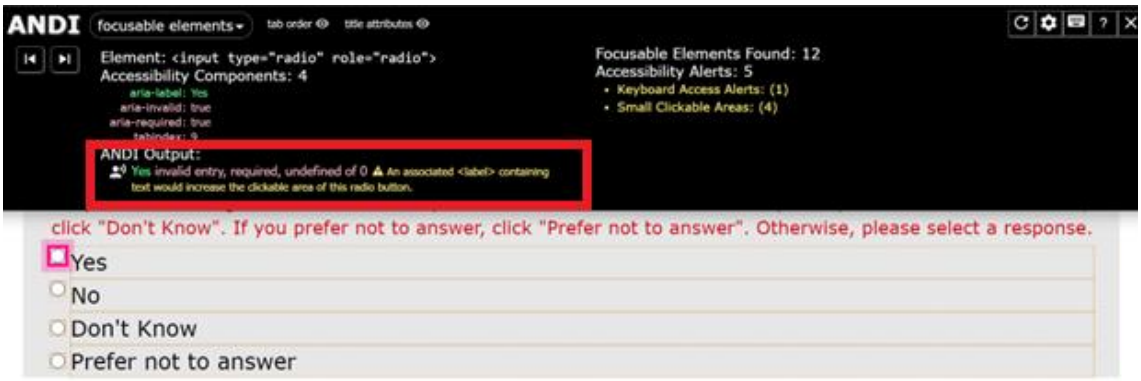
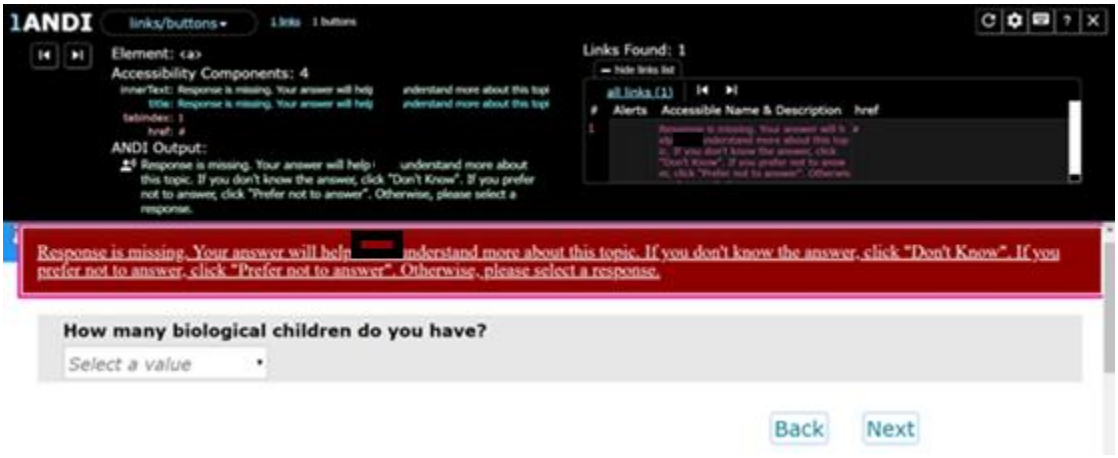


Figure 4b. Text on the Page Is Denoted as a Link, but It Is Not Visible; The Element Is a Hidden Error Message



Unfortunately, we were unable to find a way to fix item #1 (Figure 4a) in Blaise 5.12.8.

Oddly enough, we were able to fix item #2 (Figure 4b) by *turning off* some of the built-in Blaise 5 accessibility settings. By unchecking the "Use Skip Links" option under "Accessibility Options" on the "Data Entry" tab under "Settings," this 508-compliance failing was corrected.

5. Full Instrument Testing

Similar documents were produced for 508-compliance testing on the full instrument. Due to time constraints, the 508-compliance testing occurred as soon as the error messages and screen layouts were finished.

Testing on the full instrument revealed issues similar to those discovered during testing of the smaller instrument, as well as additional problems. While developing the test project, we had not known our final instrument would contain some large tables. These large tables presented multiple 508-compliance issues, but it was determined that the benefits of using the table outweighed the disadvantages of not being able to achieve full 508 compliance on these screens. The decision was made to keep the tables.

Issues that were already discovered during initial testing continued to cause problems in the full instrument. While design changes were considered to remedy these issues, in the end, it was determined that remediation was the better option. Rather than risk compromising the interview experience for the majority, we decided the option to complete the survey via CATI mode provided a suitable alternative for any users experiencing accessibility issues.

6. Compliance Issues That Could Not Be Solved

6.1 Enumerated Fields—Checked/Unchecked Responses

One of the first issues encountered involved all enumerated fields, which were quite numerous in the survey. WCAG 2.0 Success Criterion (SC) 1.3.1 requires programmatic labels that are meaningful and accurate. Unfortunately, for Blaise 5, we were unable to find a way to mark this information in a way that ANDI would recognize. In this case, the NVDA screen reader would read the correct responses, regardless of this ANDI-detected issue.

Figure 6a. Blaise 5 Correctly Indicates "Checked" for the Selected Response but Does *Not* Indicate "Unchecked" (aria-checked: false) for the Unselected Response and Does Not Include the Correct Number of Response Options

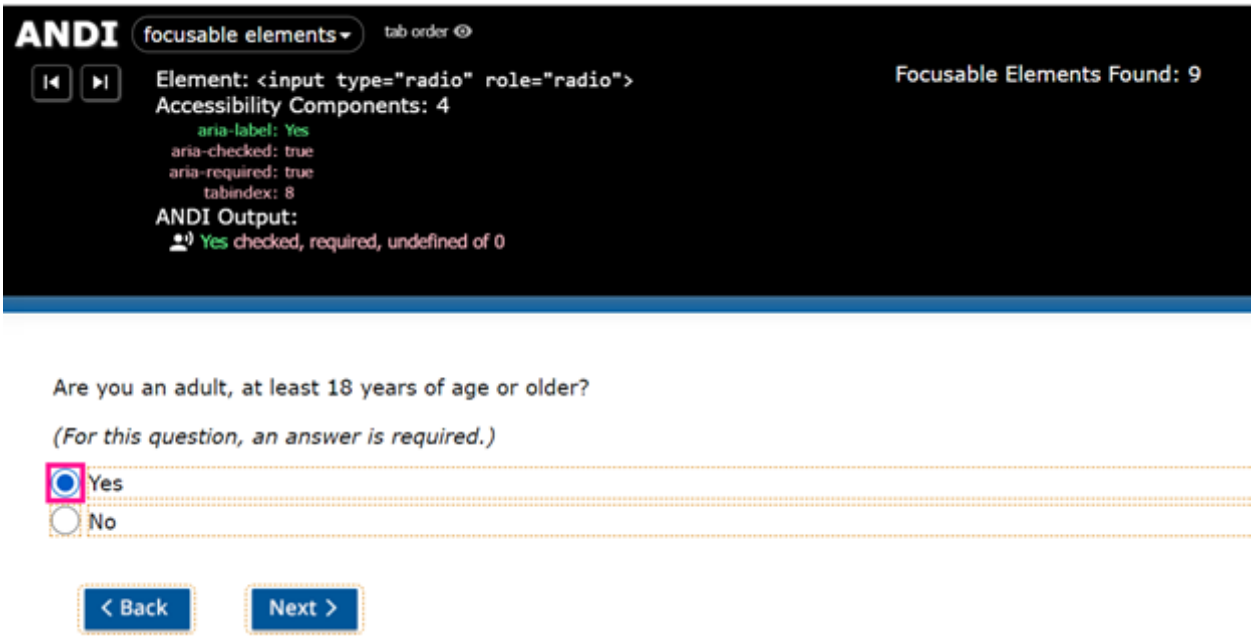


Figure 6b. Blaise 5 Does Not Indicate "Unchecked" for Responses That Are *Not* Selected

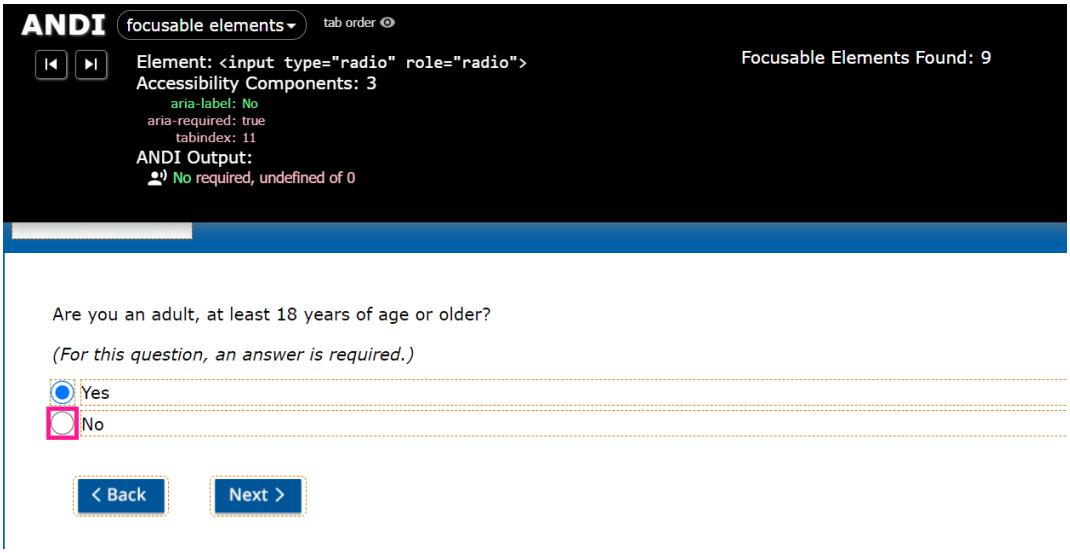


Figure 6c. An Example of a Radio Button Set That Correctly Indicates Checked/Unchecked and the Number of Response Options Available for a Selected Response

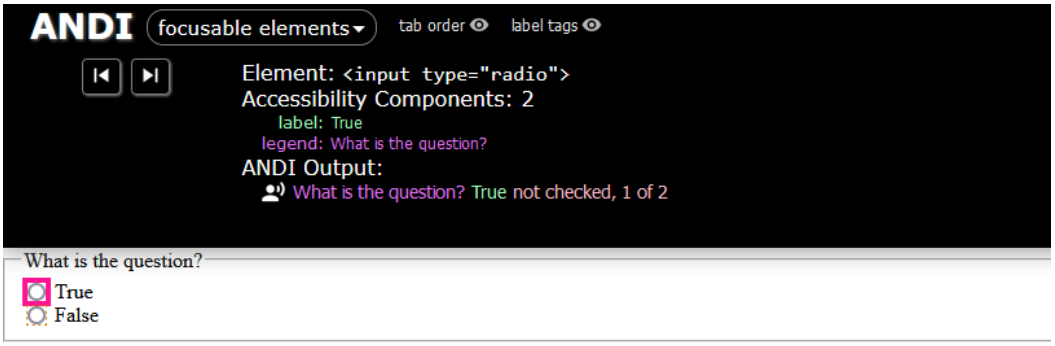
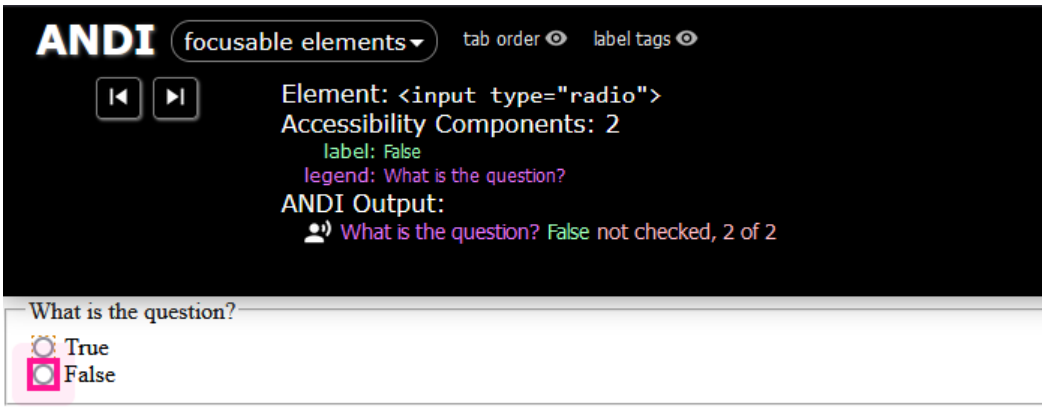


Figure 6d. An Example of a Radio Button Set That Correctly Indicates Checked/Unchecked and the Number of Response Options Available for an Unselected Response

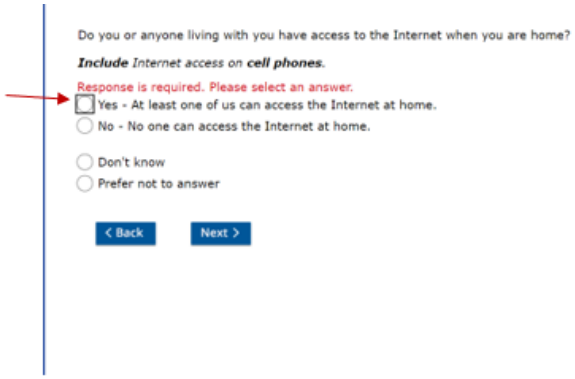


6.2 Focus Order Reveal—Cursor Does Not Move to Special Answer Response Options

SC 2.0–2.4.3 of WCAG requires the focus to automatically move to additional content that is revealed. In this particular survey, "Don't Know" and "Prefer Not to Answer" (DK/RF) special answer options are not initially visible. These options appear at the bottom of the valid options list only if the respondent attempts to leave the page without providing a valid response. Unfortunately, Blaise 5 does not allow for adjustments to the focus order and refreshes the page when special answers are displayed using the "Hide Special Answers the first time a respondent enters a page" option, which then sends the focus back to the first response option on the page (not the newly displayed DK/RF).

As a potential workaround to this issue, we considered relocating the DK/RF options to the top of the page so that the focus would automatically shift to these options when they were displayed. However, it was determined by our survey methodologist that this positioning may be potentially detrimental to survey results, and that—along with client preference—informed the decision to keep the DK/RF options at the bottom of the page.

Figure 6e. Displaying the DK/RF Options after the Other Response Options Caused Issues with the Focus Order Reveal Requirement, but Blaise 5 Did Not Provide Options for Adjusting the Focus Order



Notice: The system remains on the same questions and displays two additional responses, i.e., Don't Know and Prefer not to answer. The cursor focus moves to the 'Yes' response, instead of 'Don't Know' response as specified by the WCAG 2.0 – 2.4.3 Focus Order Reveal requirement.

6.3 Reading Order of the Content

Dropdown fields also had their own set of problems. SC 1.3.1 of WCAG 2.0 requires that the reading order of the content (in context) is correct. The meaning of the content should be preserved without CSS positioning. On dropdown fields, selected responses were lost when the "Linearize Page" option in ANDI was used. This option removes CSS positioning from the elements on the page.

While Blaise 5 *does* have an option to "Optimize for readability without stylesheets" that may have helped with this issue, we were unable to utilize it. Whenever we turned this option on, it caused text formatting issues for the response options (see Figure 6f below for an example), changing the text on the response options to 12pt font when they really should have matched the question text at 20pt font.

Figure 6f. Turning on the "Optimize for Readability without Stylesheets" Caused the Response Option Text to Become Tiny (the Response Options Text Size Should Match the Size of the Question Text)

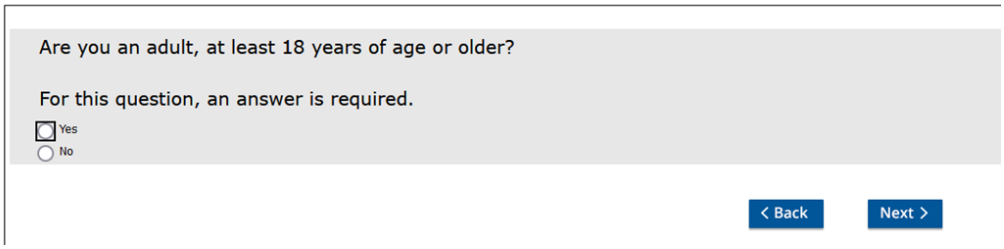
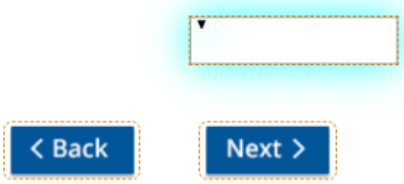


Figure 6g. The Selected Response Option Is Visible Prior to Utilizing the "Linearize Page" Option in ANDI

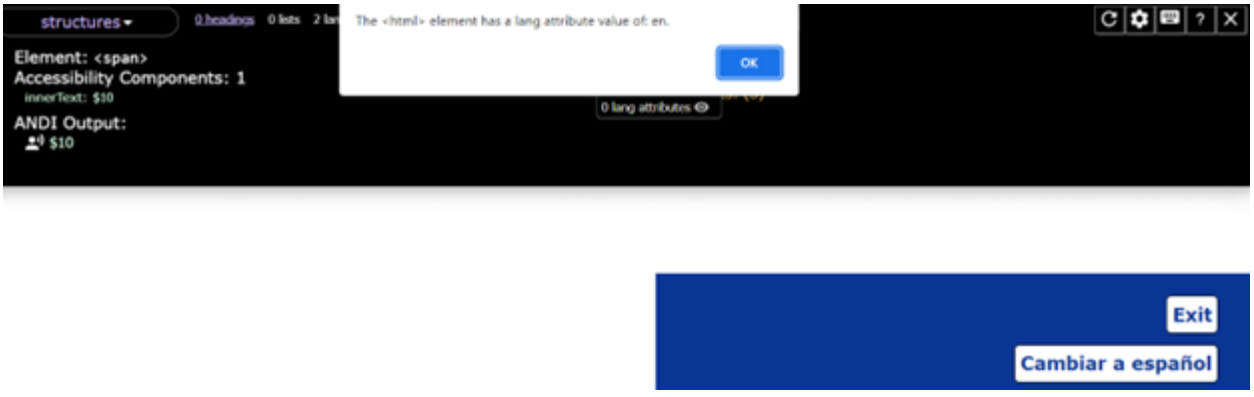
Figure 6h. When the "Linearize Page" Option Is Turned on, the Selected Response and Associated Label Disappear from the Screen



6.4 Part Language Defined

The overall page language is defined by Blaise 5, but we were unable to set the language on individual parts of a page. For example, the language selector was displayed in the opposite language than the one used on the page and, as such, should have been defined separately from the rest of the page. This definition is a requirement in SC 3.1.2 of WCAG 2.0.

Figure 6i. The Page Is Defined in English (en) and There Are No Additional Language Tags Defined (Shown Here as "0 lang attributes"), but There *Should* Be One Defined for the "Cambiar a español" Button



The Blaise 5 Help suggested naming individual parts on a page by setting the language tag for rich text elements. For example, placing text between these starting and ending tags, <lang value=”fr”></lang>, would specify that the enclosed text is in French. We attempted to implement these tags for the select language option, but Blaise 5’s <lang> tags are not recognized by ANDI. Regardless of the lack of recognition by ANDI, we left the <lang> tags in the final survey in case they are recognized by some accessibility software.

6.5 Font Enhancements and HTML Links

In traditional HTML, and <emphasis> tags should be used to bold or italicize text in a way that promotes accessibility. Blaise 5, on the other hand, utilizes its own set of tags to denote bold or italicized text. Unfortunately, the Blaise 5 method of bolding text is recognized as a “heading” by ANDI, which causes issues. WCAG 2.0 SC 1.3.1 specifies that every heading that can be programmatically determined is a visual heading, and that every visual heading can be programmatically determined. We tried utilizing /<emphasis> tags, but they were not recognized by Blaise 5 as marking text that should be bolded/italicized.

Figure 6j. Text Bolded Using Blaise 5 Tags Is Recognized as a Heading by ANDI

A similar issue occurred with HTML links. In standard HTML, links are designated using <a href> tags, but Blaise 5 utilizes its own <hyperlink> tag. This hyperlink tag is not recognized by ANDI as a link or focusable element. This lack of recognition causes issues with WCAG 2.0 SC 2.4.4, which requires the purpose of each link to be able to be determined from the link text alone or by both the link text and programmatically determined link context.

7. Workarounds to Correct Compliance Issues

7.1 Constraint Errors

While there were many issues that we could not find a way to circumvent, we did find workarounds for some of the accessibility issues. One of the issues discovered by our accessibility testers was that a respondent could not move backwards after receiving a Blaise 5 constraint error. We were using constraint errors to prevent respondents from entering invalid email addresses (as suggested by the Blaise 5 Help). To get around this issue, we developed a procedure to check email addresses. This procedure exports an "IsValid" value that evaluates to '1' when the email is valid. By utilizing a signal and this procedure, we were able to allow the respondent to move backwards in the survey, even if they had entered an invalid email address (they still could not move forward).

Figure 7a. The CheckEmailProcedure Evaluates the Entered Email Address to Check Its Validity and Will Return a 1 (valid) or 0 (invalid)

```
PROCEDURE CheckEmailAddress
  PARAMETERS
    IMPORT
      EmailAddress : STRING
    EXPORT
      IsValid : 0..1
  RULES
    IsValid := 0

    IF POSITION('@', EmailAddress) > 0 AND POSITION('.', EmailAddress) > 0 THEN //contains an @ symbol and .
      IF LEN(SUBSTRING(EmailAddress, 1, POSITION('@', EmailAddress)+1)) >= 1 THEN //at least 1 character before @ symbol
        IF LEN(SUBSTRING(EmailAddress, POSITION('@', EmailAddress)+1, POSITION('.', EmailAddress))) >= 1 THEN //at least 1 character between @ symbol and .
          IF LEN(SUBSTRING(EmailAddress, POSITION('.', EmailAddress)+1, LEN(EmailAddress))) >= 1 THEN // at least 1 character after .
            IsValid := 1
          ENDIF
        ENDIF
      ENDIF
    ENDIF
ENDPROCEDURE
```

Figure 7b. This Signal Will Prevent the Respondent from Moving Forward If the Email Address Is Invalid, but Still Allows the Respondent to Go Backwards in the Survey

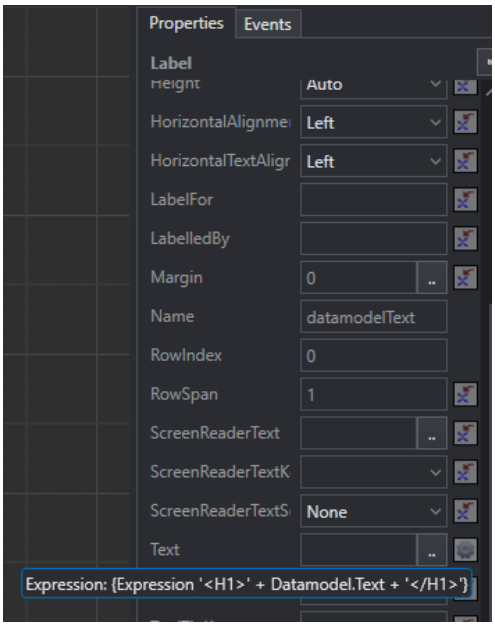
```
SIGNAL ( EMAILFIELD <> EMPTY AND IsEmailValid = 1 )
  ENG "Email address is invalid. Please provide a valid email address (abc@xyz.com)."
  SPA "La dirección de correo electrónico no es válida. Proporcione una dirección de correo electrónico válida (abc@xyz.com)."
```

7.2 Page Title Is Not Programmatically Identified as a Heading

Another issue identified was that the page title, which was the data model name, was not identified as a heading. This lack of identification conflicted with the requirements of WCAG 2.0 SC 1.3.1, "Info and Relationships" that requires information and relationships to be able to be programmatically determined.

Adding the <H1> tag to the data model name in the Blaise 5 source code would have fixed this issue, but then it also would have been applied in both layout sets, large and small. Adding the <H1> tag overrides any other font settings and modifying it in the Blaise source would make the data model name H1-sized text in the small layout set, as well. To avoid making the data model name this large in the small layout set, we adjusted the page header template part used only in the large layout set. The small layout set, which we did *not* evaluate for 508 compliance, retained the small text determined to look best in the small layout set.

Figure 7c. Adding <H1> Tags on Text for the Label to Display the Data Model Name Corrected This Issue While Allowing the Small Layout Set to Retain the Smaller Font Size; This Change Was Made on the Template Part Used for the Header in the Large Layout Set



7.3 Tables

Errors in tables presented another accessibility challenge. WCAG 2.0 SC 3.3.1 requires errors to be clearly identified. Part of that requirement is that the error message reference the associated field. Blaise 5 tables initially did not fulfill this requirement, but we were able to modify the templates to display information that would make these tables 508-compliant. This solution was not implemented in the final production survey, however, because it was decided displaying field names (as required for accessibility compliance) might prove confusing for most respondents.

Figure 7d. Default Templates Do Not Make a Reference to the Field Name, Either Where the Question Is Displayed or in the Error Message When It Displays

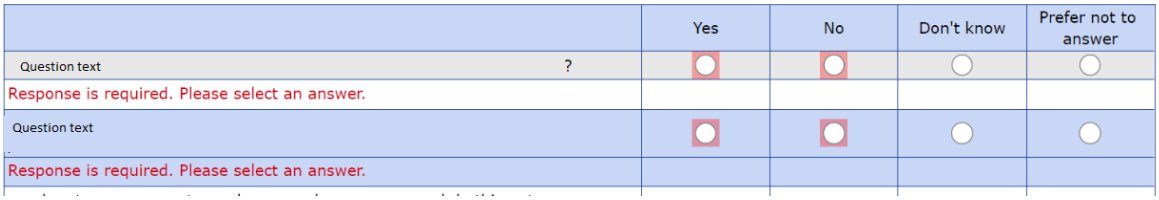
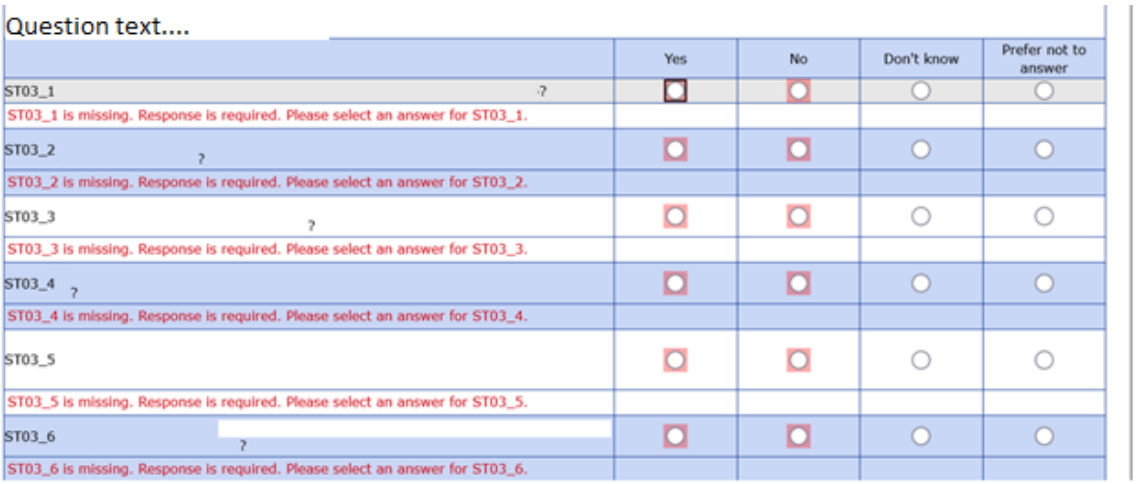


Figure 7e. Adjusted Templates Display the Field Name with the Question Text and in the Error Message When It Is Displayed so the Respondent Can Easily Tell Which Error Applies to Which Row in the Table



7.4 Magnification to 200%

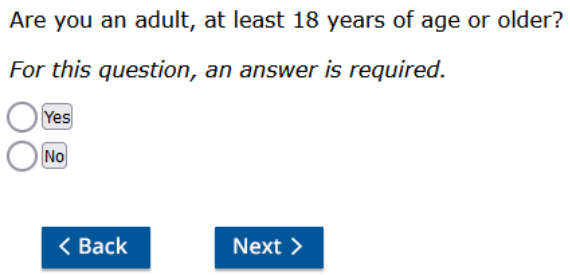
When size (pixels, in this case) is used to determine the layout set, Blaise 5 reevaluates the layout set used when the respondent magnifies the browser window. This feature caused us issues with WCAG 2.0 SC 1.4.4, which requires content to be able to scale up to 200% without changes to the content on screen. Our testing team determined that in the process of scaling the content up to 200%, the page would update from using the large layout set to the small layout set. To work around this issue, we adjusted the condition used to determine when to switch from the small layout set to the large one. Instead of using 800 px as the cutoff for the switch from the large layout set to the small one, we used 600 px. This cutoff point kept the layout set used from updating when scaling up from 100% to 200%.

8. Communication with the Blaise Development Team

As always, the Blaise development team was very responsive to our accessibility questions and did their best to help us find ways to circumvent the issues identified. Some of the issues we reported were identified as bugs and corrected in a subsequent build, but due to time constraints for testing, we determined it would not be worth the risk to switch major versions so close to the launch date for the instrument. Other issues were corrected quickly enough that we could utilize the newer build prior to production. Initially, we experienced a problem where the tab order would jump to an incorrect spot after selecting a response from a dropdown. The Blaise development team determined this issue was a bug and it was promptly fixed in the subsequent build, 5.12.8, which we used in production for this instrument.

A few of the issues we reported were “fixable” by turning on some of the Blaise 5 accessibility options in the BLAX “Settings” tab, but these options caused other issues. For an issue where the error message was not receiving focus correctly, it was suggested to turn on the “Use CSS grids instead of the size tree” option on the settings tab of the BLAX to correct it. However, turning on this option caused the response options to display as buttons.

Figure 8a. Turning on the "Use CSS Grids Instead of the Size Tree" Option Caused Response Options to Appear as Buttons



Other issues were identified by ANDI as problems but worked correctly in NVDA. The issue with radio buttons identifying as "0 of undefined" is an example of a problem that showed in ANDI but worked correctly in NVDA. The Blaise development team said that NVDA read out (correctly) "4 of 4" for a field with four radio buttons, while ANDI still saw it as "0 of undefined."

9. Challenges in 508-Compliance Testing

In addition to trying to navigate the sometimes complex 508-compliance requirements, we also had to contend with bugs found in the ANDI software while testing. One of the compliance issues discovered by our compliance testing team was "WCAG 2.0–2.4.4 Link Purpose: The purpose of each link or button can be determined from any combination of the link/button text." We discovered that ANDI had a bug: it wouldn't detect the Blaise 5 buttons when the screen was initially loaded. Instead, it required clicking on the "0 buttons" link on the page.

Figure 9a. Upon Initially Loading the Page Where the 508-Compliance Error Was Identified, ANDI Showed "0 buttons"; This Link Is Identified in the Image by the Large Red Arrow

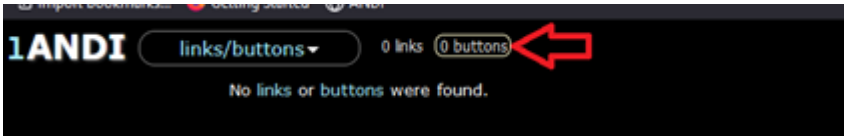
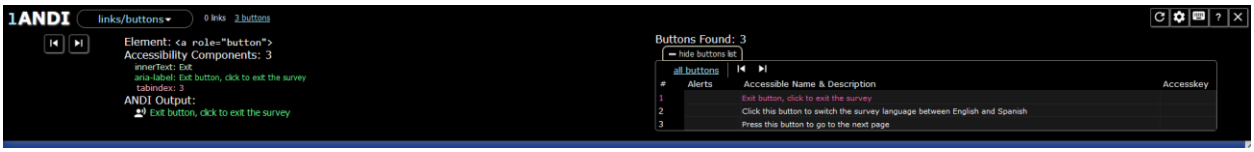


Figure 9b. After Clicking on the Previously Mentioned "0 buttons," ANDI Correctly Identified All Blaise 5 Buttons on the Page as Buttons



As mentioned previously, sometimes we were able to achieve better compliance by turning *off* certain Blaise 5 "accessibility" features. While it seems odd that *not* using features specifically meant to facilitate accessibility improved our accessibility testing "score," we believe this apparent conflict is the result of different methods of 508-compliance testing. In our communications with the Blaise development team (discussed in more detail in Section 8), it was discovered that they primarily relied on testing using the NVDA screen reader. This feature that caused issues for ANDI testing likely fulfilled some requirement

of the NVDA tool. We highlight this difference in method here to highlight the difficulty in achieving total 508 compliance when different tools or methods are used to determine compliance.

To further emphasize this point, when the client conducted their own 508-compliance testing on the finished survey, they found an issue not previously identified during ANDI testing. Not all WCAG 2.0 requirements can be measured using the ANDI tool, so individual tester evaluation and analysis can add yet another layer of variability to compliance testing. 508-compliance testing results will vary as much as the tools, methods, and testers used to conduct it.

10. Final Results and Future Plans

We were unable to achieve full 508 compliance using Blaise 5.12.8. Fortunately, we were able to resolve all compliance issues identified as “severe” by the client and only had a small number of more minor issues left unresolved. Bugs identified during testing for this instrument should ideally be corrected in future versions of Blaise 5, where possible. The testing process also taught us several valuable lessons that helped us get a few steps closer to 508 compliance, and we hope that future versions of Blaise 5 will allow us to get even closer.

11. References

ANDI (Accessibility Testing Tool), Accessible Solutions Branch of the Social Security Administration
https://www.ssa.gov/accessibility/andi/help/install.html

Blaise 5 Help Reference Manual, Statistics Netherlands
http://help.blaise.com/

WCAG 2.1 Understanding Docs, W3C Web Accessibility Initiative (WAI)
https://www.w3.org/WAI/WCAG21/Understanding/