W3C Group Note

Guidance on Applying WCAG 2 to Non-Web Information and Communications Technologies (WCAG2ICT)

W3C Group Note 11 December 2025

▼ **More details about this document**

This version:
https://www.w3.org/TR/2025/NOTE-wcag2ict-22-20251211/

Latest published version:
https://www.w3.org/TR/wcag2ict-22/

Latest editor's draft:
https://w3c.github.io/wcag2ict/

History:
https://www.w3.org/standards/history/wcag2ict-22/
Commit history

Editors:
Mary Jo Mueller (IBM)
Phil Day (NCR Atleos)
Daniel Montalvo (W3C)

Former editors:
Michael Cooper (W3C)
Peter Korn (Amazon)
Chris Loiselle (Oracle Corporation)
Andi Snow-Weaver (IBM)
Gregg Vanderheiden (Invited Expert, Trace Research and Development Center)

Feedback:
GitHub w3c/wcag2ict (pull requests, new issue, open issues)

21 August 2025 Version
https://www.w3.org/TR/2025/NOTE-wcag2ict-22-20250815/

15 November 2024 Version
https://www.w3.org/TR/2024/NOTE-wcag2ict-22-20241115/

WCAG2ICT 2.0 W3C Group Note

https://www.w3.org/TR/wcag2ict-20/

Abstract

This document describes how the Web Content Accessibility Guidelines (WCAG) versions 2.0 [*WCAG20*], 2.1 [*WCAG21*], and 2.2 [*WCAG22*] principles, guidelines, and success criteria can be applied to non-web Information and Communications Technologies (ICT), specifically to non-web documents and software. It provides informative guidance (guidance that is not normative and does not set requirements).

This document is part of a series of technical and educational documents published by the W3C Web Accessibility Initiative (WAI) and available from the WCAG2ICT Overview.

Status of This Document

This section describes the status of this document at the time of its publication. A list of current W3C publications and the latest revision of this technical report can be found in the W3C standards and drafts index.

This is a W3C Group Note on Applying WCAG 2 to Non-Web Information and Communications Technologies (WCAG2ICT). The purpose of this work is to update the previous WCAG2ICT Note's guidance to include changes made in WCAG 2.1 and 2.2, as well as to address public feedback received after first publication of this updated Note.

To comment, file an issue in the W3C WCAG2ICT GitHub repository. Create separate GitHub issues for each topic, rather than commenting on multiple topics in a single issue. It is free to create a GitHub account to file issues. If filing issues in GitHub is not feasible, send email to public-wcag2ict-comments@w3.org (comment archive).

This document was published by the Accessibility Guidelines Working Group as a Group Note using the Note track.

This Group Note is endorsed by the Accessibility Guidelines Working Group, but is not endorsed by W3C itself nor its Members.

The W3C Patent Policy does not carry any licensing requirements or commitments on this document.

This document is governed by the 18 August 2025 W3C Process Document.

Table of Contents

§ Introduction

§ Background

This document is an update to a W3C Working Group Note to incorporate new guidelines, success criteria, and definitions added in WCAG 2.1 and 2.2.

Guidance on Applying WCAG 2.0 to Non-Web Information and Communications Technologies (WCAG2ICT), approved in September 2013, described how WCAG 2.0 could be applied to non-web documents and software. WCAG2ICT was organized to mirror WCAG's sections: Perceivable, Operable, Understandable, and Robust. WCAG2ICT clarified when and how WCAG success criteria could be applied to non-web documents and software. Some were applicable without modification and

some were applicable with edits and/or notes. Glossary terms were also reviewed. Level AAA success criteria were not addressed in the 2013 WCAG2ICT Working Group Note.

The 2013 version of WCAG2ICT has been relied upon in regulations and legislation. An example is [*etsi-en-301-549*] (Europe) as well as other standards that reference or incorporate EN 301 549 (e.g., India, Kenya, Australia). Another example is the U.S. Section 508's Application of WCAG 2.0 to Non-Web ICT, where WCAG was incorporated by reference into Section 508 as the accessibility standard applicable to non-web documents and which also requires WCAG conformance for non-web software.

These standards looked to WCAG2ICT for detailed direction and guidance on how to apply WCAG non-web technology. The WCAG2ICT guidance also led to a few exceptions where specific success criteria are not required in non-web contexts.

§ Guidance in This Document

This document provides informative guidance (guidance that is not normative and does not set requirements) with regard to the interpretation and application of Web Content Accessibility Guidelines (WCAG) to non-web information and communications technologies (ICT). This document is a Working Group Note (in contrast to WCAG 2.0, WCAG 2.1, and WCAG 2.2, which are W3C Recommendations). Specifically, this document provides informative guidance on applying WCAG 2.0, 2.1, and 2.2 Level A and AA success criteria to non-web ICT, specifically to non-web documents and software.

> NOTE 1
> Hereafter, the use of WCAG 2 means all WCAG 2.x versions — 2.0, 2.1, and 2.2.

This document is intended to help clarify how to use WCAG 2 to make non-web documents and software more accessible to people with disabilities. Addressing accessibility involves addressing the needs of people with auditory, cognitive, neurological, physical, speech, and visual impairments, as well as the accessibility needs of people caused by the effects of aging. Although WCAG 2 addresses some user needs for people with cognitive and learning disabilities, as well as those with mental health-related disabilities, following the WCAG supplement Making Content Usable for People with Cognitive and Learning Disabilities is recommended for non-web ICT in order to address the user needs of these groups. Developers are also encouraged to obtain testing input from people with disabilities who use their applications and content.

Although this document covers a wide range of issues, it is not able to address all the needs of all people with disabilities. Since WCAG 2 was developed for the Web, addressing accessibility for non-

web documents and software may involve requirements and considerations beyond those included in this document. Authors and developers are encouraged to seek relevant advice about current best practices to ensure that non-web documents and software are as accessible as possible to people with disabilities. The following supporting documents contain helpful information for learning about the user needs, intent, and generalized implementation techniques to support a wider range of people with disabilities:

- WCAG 2 Overview
- Techniques for WCAG 2.2 [*WCAG22-TECHS*]
- How to Meet WCAG (Quick Reference)
- Additional Accessibility Guidelines Working Group - Publications

While WCAG 2 was designed to be technology neutral, it assumes the presence of a “user agent” such as a browser, media player, or assistive technology that is used as a means to access web content. As a result, the application of WCAG 2 to documents and software in non-web contexts necessitates some interpretation in order to determine how the intent of each WCAG 2 success criterion could be met in these different contexts of use. Therefore, the bulk of the WCAG2ICT Task Force's work involved evaluating how each WCAG 2 success criterion would apply in the context of non-web ICT.

The WCAG2ICT Task Force found that the majority of WCAG 2 success criteria can be applied to non-web documents and software with either no or minimal changes. Since many of the Level A and Level AA success criteria do not include any web-related terms, they apply directly as written and as described in the “Intent” sections from the Understanding WCAG 2.2 [*UNDERSTANDING-WCAG22*] resource. Additional notes were provided, as needed, to increase understanding about applying WCAG success criteria to non-web documents and software.

§ Interpretation of Web Terminology in a Non-Web Context

When certain web-specific terms or phrases like “web page(s)” were used in success criteria, those were replaced with non-web terms or phrases like “non-web document(s) and software”. Additional notes were also provided to explain the terminology replacements.

A small number of success criteria are written to apply to “a set of web pages” or “multiple web pages” and depend upon all pages in the set to share some characteristic or behavior. Since the unit of conformance in WCAG 2 is a single web page, the task force agreed that the equivalent unit of conformance for non-web documents is a single document. It follows that an equivalent unit of evaluation for a “set of web pages” would be a “set of documents”. Since it isn't possible to

unambiguously carve up non-web software into discrete pieces, a single “web page” was equated to a “software program” and a “set of web pages” was equated to a “set of software programs”. Both of these terms are defined in the Key Terms section of this document. See “set of documents” and “set of software programs” to determine when a group of documents or pieces of software are considered a set.

> NOTE
> Sets of non-web software that meet this definition appear to be extremely rare.

Not all success criteria have been fully adopted in all local regulations and legislation, and may not be applicable to all technologies. WCAG2ICT has been used in some regulations to determine whether or not to apply certain success criteria. For example, some local standards such as Section 508 in the U.S., and EN 301 549 in Europe, state that WCAG 2.0 Success Criteria 2.4.1 Bypass Blocks, 2.4.5 Multiple Ways, 3.2.3 Consistent Navigation, and 3.2.4 Consistent Identification do not apply to non-web documents and non-web software. In addition, EN 301 549 states that 2.4.2 Page Titled and 3.1.2 Language of Parts do not apply to non-web software. In contrast, the U.S. Department of Justice regulation, Nondiscrimination on the Basis of Disability; Accessibility of Web Information and Services of State and Local Government Entities (89 FR 31320, 24 April 2024), directs implementers to utilize the guidance in this document to determine the applicability of success criteria and how to apply the requirements to mobile applications. Since this document does not specifically say which criteria can or should apply, those implementing this document (WCAG2ICT) should consider the applicability of individual success criteria to non-web documents and software.

The glossary terms in WCAG 2 were also reviewed and most of them applied to non-web documents and software, as written. Some applied with additional notes or edits (largely related to phrases like “web page(s)”), and a small number of terms were only used in Level AAA success criteria, which are not addressed by the WCAG2ICT Note at this time.

§ Excluded from Scope

The following are out of scope for this document:

- This document does not seek to determine which WCAG 2 provisions (principles, guidelines, or success criteria) should or should not apply to non-web documents and software, but rather — if applied — *how* they would apply.
- This document does not propose changes to WCAG 2 or its supporting documents. However, during the development of this document, the WCAG2ICT Task Force did seek clarification on

the intent of a number of the success criteria, which led to clarifications in the Understanding WCAG 2 document.

- This document does not include interpretations for implementing WCAG 2 in web technologies.
- This document is not sufficient by itself to ensure accessibility in non-web documents and software. This is because this document does not address accessibility requirements beyond those covered by WCAG which, as a web standard, does not fully cover all accessibility requirements for non-user interface aspects of platforms, user-interface components as individual items, or ICT with closed functionality (where there is no assistive technology to communicate programmatic information).
- This document does not comment on hardware aspects of products because the basic constructs on which WCAG 2 is built do not apply to hardware.
- This document does not provide supporting techniques for implementing WCAG 2 in non-web documents and software.
- This document is purely an informative Note about non-web ICT. It is not a standard — so it does not describe how non-web ICT should conform to it.

§ Document Overview

This document includes text quoted from the WCAG 2.2 principles, guidelines, success criteria, and glossary definitions without any changes. The guidance provided by this document for each principle, guideline, success criterion, and definition is preceded by a heading beginning with “Applying…”. This guidance was created by the WCAG2ICT Task Force, then reviewed and approved by the Accessibility Guidelines Working Group.

§ Document Conventions

The following stylistic conventions are used in this document:

- Quotes from WCAG 2 are in `<blockquote>` elements and visually styled with a gray bar on the left, and immediately follow the heading for the principle, guideline, or success criterion.
- Additional guidance provided by this document begins with the phrase “Applying” and has no special visual styling.

- Replacement text that is presented to show how a success criterion would read as modified by the advice in this document are in `<ins>` elements that are visually styled as green text with a dotted underline.
- Links that are contained in the word replacement text are also styled with bold text.
- Notes are slightly inset and begin with the phrase “NOTE” — each note is in its own inset box that is styled in pale green with a darker green line on the left side of the box.
- Where WCAG Notes have been replaced or significantly rewritten in the guidance, they are notated with "(REPLACED)"; and where WCAG2ICT added new notes, they are notated with " (ADDED)".
- References to glossary items from WCAG 2 are presented in `<cite>` elements that are visually styled as ordinary text with a dotted underline, and contain title attributes noting these are WCAG definitions — they turn blue with a yellow background when mouse or keyboard focus is placed over them.
- References to glossary items in this document are presented in `<cite>` elements that are visually styled as ordinary text with a dark gray underline.
- Hereafter, the short title “WCAG2ICT” is used to reference this document.
- In headings, the term "Success Criterion" has been shortened to “SC” for brevity.

§ Comparison with the 2013 WCAG2ICT Note

The following changes and additions have been made to update the 2013 WCAG2ICT document to incorporate the new features in WCAG 2.1, the new features in WCAG 2.2, and the change to 4.1.1 Parsing listed in the Comparison with WCAG 2.1 section:

- New Background section to explain the history and known uses of WCAG2ICT
- New key terms introduced by WCAG2ICT:
  - closed functionality
  - menu-driven interface
  - platform software
  - virtual keyboard
- New WCAG 2.1 success criteria and guidelines:
  - Success Criterion 1.3.4 Orientation

    - Success Criterion 1.3.5 Identify Input Purpose
    - Success Criterion 1.4.10 Reflow
    - Success Criterion 1.4.11 Non-text Contrast
    - Success Criterion 1.4.12 Text Spacing
    - Success Criterion 1.4.13 Content on Hover or Focus
    - Success Criterion 2.1.4 Character Key Shortcuts
    - Guideline 2.5 Input Modalities
    - Success Criterion 2.5.1 Pointer Gestures
    - Success Criterion 2.5.2 Pointer Cancellation
    - Success Criterion 2.5.3 Label in Name
    - Success Criterion 2.5.4 Motion Actuation
    - Success Criterion 4.1.3 Status Messages
- New WCAG 2.2 success criteria:
    - Success Criterion 2.4.11 Focus Not Obscured (Minimum)
    - Success Criterion 2.5.7 Dragging Movements
    - Success Criterion 2.5.8 Target Size (Minimum)
    - Success Criterion 3.2.6 Consistent Help
    - Success Criterion 3.3.7 Redundant Entry
    - Success Criterion 3.3.8 Accessible Authentication (Minimum)
- Obsolete and Removed WCAG 2.2 success criteria that have errata for WCAG 2.0 and 2.1:
    - Success Criterion 4.1.1 Parsing
- New terms from WCAG 2.1 and 2.2:
    - added to Glossary Items that Apply to All Technologies:
        - dragging movements, focus indicator, minimum bounding box, pointer input, process, single pointer, state, and status message
    - added to Glossary Items Used only in AAA Success Criteria:
        - motion animation, perimeter, region, and user inactivity
    - added to Glossary Items with Specific Guidance:
        - cognitive function test
        - css pixel

    - down event
    - keyboard shortcut
    - style property
    - target
    - up event
- Updated terms:
    - large scale
    - set of web pages
    - set of non-web documents
    - set of software programs
- Updated sections:

> NOTE
>
> - All of the existing sections have undergone WCAG2ICT Task Force review to identify the necessary updates.
> - Many sections needed only minor editorial and link URL updates, such as the guidance for the WCAG 2.0 success criteria.

§ Key Terms

WCAG2ICT provides some key glossary terms to address differences between web and non-web contexts and to introduce terms that are nonexistent in WCAG but important to define for a non-web context. “Content” and “user agent” are glossary terms from WCAG 2 that need to be interpreted significantly differently when applied to non-web ICT. The glossary term “web page” in WCAG 2 is replaced with the defined terms “document” and “software”, and both “set of web pages” and “multiple web pages” are replaced with the defined terms “set of documents” and “set of software programs”. The terms introduced by WCAG2ICT are “accessibility services of platform software” because non-web software doesn't leverage the WCAG notion of a user agent, and "closed functionality" which is specific to non-web software. The remaining glossary terms from WCAG 2 are addressed in Chapter 7 Comments on Definitions in WCAG 2 Glossary. Terms defined and used in WCAG2ICT are applicable only to the interpretation of the guidance in this document. The particular

definitions should not be interpreted as having applicability to situations beyond the scope of WCAG2ICT. Further information on usage of these terms follows.

§ Accessibility Services of Platform Software

The term **accessibility services of platform software**, as used in WCAG2ICT, has the meaning below:

accessibility services of platform software (as used in WCAG2ICT)
: services provided by an operating system, user agent, or other platform software that enable non-web documents or software to expose information about the user interface and events to assistive technologies and accessibility features of software

> NOTE
>
> These services are commonly provided in the form of accessibility APIs (application programming interfaces), and they provide two-way communication with assistive technologies, including exposing information about objects and events.

§ Closed Functionality

The term **closed functionality**, as used in WCAG2ICT, has the meaning below:

closed functionality (as used in WCAG2ICT)
: a property or characteristic that prevents users from attaching, installing, or using assistive technology

> NOTE 1
>
> To support users with disabilities, ICT with closed functionality might instead provide built-in features that function as assistive technology or use other mechanisms to make the technology accessible.

Example: Examples of technology that may have closed functionality include but are not limited to:

- self-service transaction machines or kiosks — examples include machines used for retail self-checkout, point of sales (POS) terminals, ticketing and self-check-in, and Automated Teller Machines (ATMs).
- telephony devices such as internet phones, feature phones, smartphones, and phone-enabled tablets
- educational devices such as interactive whiteboards and smart boards
- entertainment technologies including gaming platforms or consoles, smart TVs, set-top boxes, smart displays, smart speakers, smart watches, and tablets
- an ebook reader or standalone ebook software that allows assistive technologies to access all of the user interface controls of the ebook program (open functionality) but does not allow the assistive technologies to access the actual content of book (closed functionality).
- medical devices such as digital blood pressure monitors, glucose meters, or other wearable devices
- an operating system that makes the user provide login credentials before it allows any assistive technologies to be loaded. The login portion would be closed functionality.
- other technology devices, such as printers, displays, and Internet of Things (IoT) devices

NOTE 2

Some of these technologies, though closed to some external assistive technologies, often have extensive internal accessibility features that serve as assistive technology that can be used by applications on these devices in the same way assistive technology is used on fully open devices, such as desktop computers. Others are open to some types of assistive technology but not others.

§ Content (on and off the web)

WCAG 2 defines **content** as:

> information and sensory experience to be communicated to the user by means of a user agent, including code or markup that defines the content's structure, presentation, and interactions

For non-web content it is necessary to view this a bit more broadly. Within WCAG2ICT, the term “content” is used as follows:

content (non-web content) (as used in WCAG2ICT)
: information and sensory experience to be communicated to the user by means of **[software]**, including code or markup that defines the content's structure, presentation, and interactions

> NOTE 1
>
> Non-web content occurs in two places; documents and software. When content occurs in a non-web document, a user agent is needed in order to communicate the content's information and sensory experience to the user. When content occurs in non-web software, a separate user agent isn't needed — the software itself performs that function.

> NOTE 2
>
> Content from a third party needs special consideration since sometimes it may be under the control of the author (e.g. contracted and therefore may not be considered 3rd party) and sometimes it is completely out of the control of the author (e.g. email in an email client).

> NOTE 3
>
> For non-web software, content also includes the user interface.

> NOTE 4
>
> Within WCAG2ICT wherever “content” or “web content” appears in a success criterion it is replaced with “content” using the definition above.

§ Document

The term **document**, as used in WCAG2ICT, has the meaning below:

document (as used in WCAG2ICT)

assembly of content, such as a file, set of files, or streamed media that functions as a single item rather than a collection, that is not part of software and that does not include its own user agent

> NOTE 1
>
> A document always depends upon a user agent to present its content to the user.

> NOTE 2
>
> Letters, spreadsheets, emails, books, pictures, presentations, and movies are examples of documents.

> NOTE 3
>
> Software configuration and storage files such as databases and virus definitions, as well as computer instruction files such as source code, batch/script files, and firmware, are examples of files that function as part of software and thus are not examples of documents. If and where software retrieves “information and sensory experience to be communicated to the user” from such files, it is just another part of the content that occurs in software and is covered by WCAG2ICT like any other parts of the software. Where such files contain one or more embedded documents, the embedded documents remain documents under this definition.

> NOTE 4
>
> A collection of files zipped together into an archive, stored within a single virtual hard drive file, or stored in a single "encrypted file system" file, do not constitute a single document.

NOTE 5

Anything that can present its own content without involving a user agent, such as a self-playing book, is not a document but is software.

NOTE 6

A single document may be composed of multiple files such as the video content, closed caption text, etc. This fact is not usually apparent to the end-user consuming the document / content. This is similar to how a single web page can be composed of content from multiple URIs (e.g. the page text, images, the JavaScript, a CSS file etc.).

Example: An assembly of files that represented the video, audio, captions, and timing files for a movie would be a document.

Counterexample: A binder file used to bind together the various exhibits for a legal case would not be a document.

§ Menu-driven Interface

The term **menu-driven interface**, as used in WCAG2ICT, has the meaning below:

menu-driven interface (as used in WCAG2ICT)
: an interface composed of menus and sub-menus which the user accesses by pressing buttons or using a touch-screen

Example: Products that have a menu-driven interface include, but are not limited to, self-service transaction machines, printers, and IP-based telephones.

§ Platform Software

The term **platform software**, as used in WCAG2ICT, has the meaning below:

platform software
: software that runs on an underlying software or hardware layer and that provides a set of software services to other software components

> NOTE 1
>
> Platform software may run or host other software, and may isolate them from underlying software or hardware layers.

> NOTE 2
>
> A single software component may have both platform and non-platform aspects.

> Example: Examples of platforms are: desktop operating systems; embedded operating systems, including mobile systems; web browsers; plug-ins to web browsers that render a particular media or format; and sets of components that allow other applications to execute, such as applications which support macros or scripting.

This definition is based on the definition of "platform software" found in [*ISO_9241-171*] and [*ISO/IEC_13066-1*].

§ Set of Documents

The term **set of documents**, as used in WCAG2ICT, has the meaning below:

set of documents (non-web) (as used in WCAG2ICT)
: collection of **[documents]** that share a common purpose; are created by the same author, group or organization; **[are published together; and all refer to each other by name or link]**

NOTE 1

Republishing or bundling previously published documents as a collection does not constitute a set of documents.

NOTE 2

If a set is broken apart, the individual parts are no longer part of a set, and would be evaluated as any other individual document is evaluated.

Example: One example of a set of documents would be a three-part report where each part is a separate file. The table of contents is repeated at the beginning of each file to enable navigation to the other parts.

§ Set of Software Programs

The term **set of software programs**, as used in WCAG2ICT, has the meaning below:

set of software programs (as used in WCAG2ICT)

collection of **[software programs]** that share a common purpose; are created by the same author, group or organization; **[and are distributed together and can be launched and used independently from each other, but are interlinked each with every other one such that users can navigate from one program to another via a consistent method that appears in each member of the set]**

NOTE 1

Although "sets of web pages" occur frequently, "sets of software programs" appear to be extremely rare.

NOTE 2

Redistributing or bundling previously distributed software as a collection does not constitute a set of software programs.

NOTE 3

Consistent does not mean identical. For example, if a list of choices is provided it might not include the name of the current program.

NOTE 4

If a member of the set is separated from the set, it is no longer part of a set, and would be evaluated as any other individual software program.

NOTE 5

Any software program that is not part of a set, per this definition, would automatically satisfy any success criterion that is specified to apply to “sets of” software (as is true for any success criterion that is scoped to only apply to some other type of content).

NOTE 6

If there is any ambiguity whether the group is a set, then the group is not a set.

NOTE 7

If there is no independent method to launch the software programs (as is common in ICT with closed functionality), those programs would not meet the definition of a “set of software programs”.

NOTE 8

Although the term “software” is used throughout this document because this would apply to stand-alone software programs as well as individual software components and the software components in software-hardware combinations, the concept of “set of software programs” would only apply (by definition) to programs that can be launched separately from each other. Therefore, in the WCAG2ICT guidance for the provisions that use the phrase “set of” (success criteria 2.4.1, 2.4.5, 3.2.3, 3.2.4, and 3.2.6), the phrase “set of software programs” is used.

Example: One example of a set of software programs would be a group of programs that can be launched and used separately but are distributed together and all have a menu that allows users to launch, or switch to, each of the other programs in the group.

Counterexamples: Examples of things that are **not** sets of software programs:

- A suite of programs for authoring different types of documents (text, spreadsheets, presentations, etc.) where the programs don't provide an explicit, consistent means to launch, or switch to, each of the other programs in the group.
- An office package consisting of multiple programs that launches as a single program that provides multiple functionalities such as writing, spreadsheet, etc., but the only way to navigate between programs is to open a document in one of the programs.
- A bundle of software programs that is sold together but the only way to navigate between the programs in the bundle is to use a platform software level menu to navigate between them (and not via a menu provided by each program that allows you to navigate to just the other programs in this bundle).
- A group of programs that was a set, but the programs have been moved to separate locations so that their “set” behaviors were disrupted and no longer work. Even though they *were* a set at one time, because they are no longer installed as a set they no longer *are* a set and would not need to meet any success criteria that apply to sets of software.

§ Software

The term **software** as used in WCAG2ICT, has the meaning below:

software (as used in WCAG2ICT)

software products, or software aspects of hardware-software products, that have a user interface and do not depend upon a separate user agent to present any of its content

> NOTE 1
>
> For software, the user interface and any other embedded content is covered by these guidelines. The software provides a function equivalent to a user agent for the embedded content.

> NOTE 2
>
> Software without a user interface does not have content and is not covered by these guidelines. For example, driver software with no user interface would not be covered.

> NOTE 3
>
> Because software with a user interface provides a function equivalent to a user agent in addition to content, the application of some WCAG 2 success criteria would be different for content embedded in software versus content in a document, where it is viewed through a separate user agent (e.g. browser, player, viewer, etc.).

§ User Agent

WCAG 2 defines **user agent** as:

> any software that retrieves and presents web content for users
>
> > Example: Web browsers, media players, plug-ins, and other programs — including assistive technologies — that help in retrieving, rendering, and interacting with web content.

For non-web ICT, “user agent” needs to be viewed differently. In WCAG 2, the term “user agent” only refers to retrieval and display of web content. For non-web ICT, the term “user agent” refers to

retrieval and display of separate content that is *not on the web*, which WCAG2ICT refers to as a "document". Within WCAG2ICT, the term "user agent" is used as follows:

user agent (as used in WCAG2ICT)
any software that retrieves and presents **[documents]** for users

> NOTE 1
>
> Software that only displays the content contained within it is not considered to be a user agent. It is just considered to be software.

> NOTE 2
>
> An example of software that is not a user agent is a calculator application that doesn't retrieve the calculations from outside the software to present it to a user. In this case, the calculator software is not a user agent, it is simply software with a user interface.

> NOTE 3
>
> Software that only shows a preview of content, such as a thumbnail or other non-fully functioning presentation, is not providing full user agent functionality.

§ Virtual Keyboard

The term **virtual keyboard**, as used in WCAG2ICT, has the meaning below:

virtual keyboard (as used in WCAG2ICT)
any software that acts as a keyboard and generates output that is treated by other software as keystrokes from a keyboard

> NOTE
>
> Eye-gaze, morse code, speech, and switches (e.g. sip-and-puff) have all been used by virtual keyboards as input that generates "keystroke" output.

§ Comments on Closed Functionality

As noted in the Introduction, WCAG 2 assumes the presence of a “user agent” such as a browser, media player, or assistive technology as a means to access web content. Many of the success criteria in WCAG 2 assume web content will be accessed by ICT where assistive technologies can be connected to it or installed on it. The assistive technologies then present the web content to people with disabilities in an accessible form.

ICT with closed functionality does not allow the use of some assistive technologies for some or all of the ICT's functions. In many cases, such ICT also lacks a “user agent” or its equivalent. To the extent the ICT is closed, following the WCAG success criteria by themselves will not ensure that non-web software is accessible. Where the wide range of assistive technologies or user agents are not available, as they are for web content, to address the intent of these success criteria, something else needs to be provided or be required to facilitate accessibility as intended by WCAG 2. It is outside the WCAG2ICT Task Force Work Statement to say what additional measures are needed, but WCAG2ICT points out which success criteria depend on assistive technologies — and therefore which success criteria would be problematic in the context of ICT with closed functionality.

> Example: In developing guidance for ICT with closed functionality, the task force has considered specific examples of ICT that historically have been partially or fully closed to assistive technologies:
>
> - self-service transaction machines or kiosks (e.g. retail self-checkout, point of sales (POS) terminals, and Automated Teller Machines (ATMs))
> - telephony devices (e.g. internet phones, feature phones, and smartphones)
> - entertainment technologies (e.g. smart TV, set-top box, smart watches)
> - ebook reader
> - computer that is locked down due to a policy so that users may not adjust settings or install software
> - other technology devices (e.g. printers, displays, and Internet of Things (IoT) devices).
>
> These examples are explained more fully in the definition of closed functionality in the Key Terms section.

> NOTE
>
> Some of these technologies, though closed to some external assistive technologies, often have extensive internal accessibility features that serve as assistive technology that can be used by applications on these devices in the same way assistive technology is used on fully open devices, such as desktop computers. Others are open to some types of assistive technology but not others.

There are existing standards that specify accessibility requirements for both hardware and software aspects of ICT with closed functionality. This document does not comment on those standards, but does note that WCAG success criteria should not be applied to hardware aspects of ICT with closed functionality. WCAG2ICT does provide considerations for applying WCAG success criteria to non-web software on ICT with closed functionality. WCAG2ICT also indicates where and why success criteria might be problematic for non-web software due to the underlying assumptions built into the WCAG success criteria. See Appendix A: Success Criteria Problematic for Closed Functionality for a list of success criteria for which this is relevant.

§ Text / Command-Line / Terminal Applications and Interfaces

Text applications are a class of software ICT that appeared decades ago, prior to the emergence of the graphical user interface (GUI) and the Web. The interface of a text application is generated using only text characters, and either a hardware terminal or a software terminal application handles the rendering of the text application—similar to how a web user agent handles the rendering of a web application. Text applications only accept text input, though some may also support the use of a mouse or other input devices. More recently, terminal applications that render text applications in the GUI may utilize spoken input through Automated Speech Recognition (ASR). Both GUI and native text environment interfaces also now commonly support word-completion prediction technologies. Command-line applications are a subset of text applications with further specific properties.

Historically, assistive technologies developed alongside text applications, making it possible for text applications to be accessible. Although there are far fewer new text applications being developed compared to new GUI or web applications, text applications remain in use today. In fact, command-line interfaces have seen a resurgence in recent years, especially in popular programming and revision-tracking environments with continued development and greater functionality. In some cases this has precipitated renewed developments in assistive technology support for text applications.

Assistive technology support continues to evolve in today's text applications. Key examples include:

- In command line interfaces (CLI), support often includes context-sensitive help, so that help output following one command argument is different from the help provided following two arguments, and different still after three arguments. This helps users be more efficient and places no new requirements on assistive technologies.
- Output options generally include machine-readable structured text formats (such as JSON), in addition to the still powerful and widely used options of input/output redirection and piping. In these scenarios the assistive technology user can make use of the same range of output options as anyone else who finds the CLI environment compelling.

As noted in Appendix B. Background on Text / Command-Line / Terminal Applications and Interfaces, applying WCAG to text / command-line applications involves understanding how text applications are rendered, how text applications have been made accessible via assistive technologies, and how to apply the concepts of “accessibility supported” and “programmatically determined” to text applications.

§ Comments on Conformance

WCAG2ICT is not a standard, so it is not possible to conform to WCAG2ICT. However, some entities may wish to use the information in WCAG2ICT to help establish standards or regulations regarding accessibility in ICT that are based on WCAG 2. While such standards or regulations will need to address matters of conformance themselves, the following notes may be of assistance to those wishing to apply WCAG 2 to non-web documents and software:

1. The WCAG 2 success criteria and the conformance requirements were designed to work together, such that the language of the success criteria is based on the nature of the conformance requirements. The choice of what level to use for a given criteria (A vs. AA vs. AAA) was further influenced by a number of factors specific to the web domain, as set forth in Understanding Levels of Conformance.
2. In the WCAG 2 conformance model, a success criterion is satisfied if the item being evaluated does not fail it. If the success criterion is in relation to something that does not exist for the item being evaluated (e.g. a success criterion is about captioning audio and there is no audio), where some might consider the criterion "not applicable", the success criterion is automatically met. This approach is central to the way the success criteria in WCAG are structured and worded.
3. WCAG 2 conformance is applied to the item being evaluated (i.e., a web page) as a whole, except when a process includes use of several items, in which case all of the items that are needed in order to complete the process must conform.

4. In WCAG 2, when conformance depends upon accessibility features of the platform (i.e., a browser for web content) or on assistive technologies, WCAG 2 requires that there are assistive technologies, etc. that work with the product (web page). That is, conformance with WCAG 2 requires that only accessibility-supported ways of using technologies are relied upon to satisfy the success criteria.
5. WCAG 2 allows information on part of a page to not conform if the same information is available elsewhere on the page in a conforming fashion. However, WCAG 2 identifies four success criteria that must be met on all areas of the page because they can interfere with the user's ability to access and use other parts of the page:
    - 1.4.2 Audio Control;
    - 2.1.2 No Keyboard Trap;
    - 2.2.2 Pause, Stop, Hide;
    - 2.3.1 Three Flashes or Below Threshold.

Also, as noted in the Introduction, it wasn't possible to unambiguously carve up software into discrete pieces, and so the unit of evaluation for non-web software is the whole software program. As with any software testing, this can be a very large unit of evaluation, and methods similar to standard software testing may be needed.

§ Comments by Guideline and Success Criterion

The sections that follow are organized according to the principles, guidelines, and success criteria from WCAG 2. The text of each success criterion from WCAG 2 is copied as quoted text. Following that, the WCAG2ICT guidance is provided. The WCAG2ICT guidance can be found in the sections where the headings begin with "Applying..." to highlight that this is the content specific to this document. Within these sections custom notes added by WCAG2ICT are marked with the text "ADDED".

§ 1. Perceivable

> Information and user interface components must be presentable to users in ways they can perceive.

Applying Principle 1 Perceivable to Non-Web Documents and Software

In WCAG 2, the Principles are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Principle 1 applies directly as written.

1.1 Text Alternatives

> Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.

Applying Guideline 1.1 Text Alternatives to Non-Web Documents and Software

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 1.1 applies directly as written.

1.1.1 Non-text Content

(Level A)

> All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below.
>
> **Controls, Input**
>
> If non-text content is a control or accepts user input, then it has a name that describes its purpose. (Refer to Success Criterion 4.1.2 for additional requirements for controls and content that accepts user input.)
>
> **Time-Based Media**
>
> If non-text content is time-based media, then text alternatives at least provide descriptive identification of the non-text content. (Refer to Guideline 1.2 for additional requirements for media.)
>
> **Test**
>
> If non-text content is a test or exercise that would be invalid if presented in text, then text alternatives at least provide descriptive identification of the non-text content.
>
> **Sensory**
>
> If non-text content is primarily intended to create a specific sensory experience, then text alternatives at least provide descriptive identification of the non-text content.
>
> **CAPTCHA**
>
> If the purpose of non-text content is to confirm that content is being accessed by a person rather than a computer, then text alternatives that identify and describe the purpose of the non-text content are provided, and alternative forms of CAPTCHA using output modes for different types of sensory perception are provided to accommodate different disabilities.
>
> **Decoration, Formatting, Invisible**
>
> If non-text content is pure decoration, is used only for visual formatting, or is not presented to users, then it is implemented in a way that it can be ignored by assistive technology.

§ Applying SC 1.1.1 Non-text Content to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.1.1.

NOTE 1 (ADDED)

CAPTCHAs do not currently appear outside of the Web. However, if they do appear, this guidance is accurate.

NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ 1.2 Time-based Media

Provide alternatives for time-based media.

§ *Applying Guideline 1.2 Time Based Media to Non-Web Documents and Software*

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 1.2 applies directly as written.

§ *1.2.1 Audio-only and Video-only (Prerecorded)*

(Level A)

> For prerecorded audio-only and prerecorded video-only media, the following are true, except when the audio or video is a media alternative for text and is clearly labeled as such:
>
> **Prerecorded Audio-only**
> An alternative for time-based media is provided that presents equivalent information for prerecorded audio-only content.
>
> **Prerecorded Video-only**
> Either an alternative for time-based media or an audio track is provided that presents equivalent information for prerecorded video-only content.

§ Applying SC 1.2.1 Audio-only and Video-only (Prerecorded) to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.2.1.

> NOTE 1 (ADDED)
>
> The alternative can be provided directly in the non-web document or software – or provided in an alternate version that satisfies the success criterion.

> NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)
>
> See also the Comments on Closed Functionality.

§ *1.2.2 Captions (Prerecorded)*

(Level A)

> Captions are provided for all prerecorded audio content in synchronized media, except when the media is a media alternative for text and is clearly labeled as such.

§ Applying SC 1.2.2 Captions (Prerecorded) to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.2.2.

> NOTE (ADDED)
>
> The WCAG 2 definition of "captions" notes that "in some countries, captions are called subtitles". They are also sometimes referred to as "subtitles for the hearing impaired". Per the definition in WCAG 2, to satisfy this success criterion, whether called captions or subtitles, they would have to provide "synchronized visual and/or text alternative for both speech and non-speech audio information needed to understand the media content" where non-speech information includes "sound effects, music, laughter, speaker identification and location".

§ *1.2.3 Audio Description or Media Alternative (Prerecorded)*

(Level A)

> An alternative for time-based media or audio description of the prerecorded video content is provided for synchronized media, except when the media is a media alternative for text and is clearly labeled as such.

§ Applying SC 1.2.3 Audio Description or Media Alternative (Prerecorded) to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.2.3.

NOTE 1 (ADDED)

Audio descriptions (also called "video descriptions", "descriptive narration", and "described videos") describe important visual information needed to understand the video content, including text displayed in the video. Where the main audio track of the video fully describes important visual information, audio descriptions would not be needed at all as the requirement would already be met. When audio descriptions are needed, one way to implement them is by providing a second audio track within the synchronized media.

NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *1.2.4 Captions (Live)*

(Level AA)

> Captions are provided for all live audio content in synchronized media.

§ Applying SC 1.2.4 Captions (Live) to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.2.4.

NOTE (ADDED)

The WCAG 2 definition of “captions” notes that “In some countries, captions are called subtitles”. They are also sometimes referred to as “subtitles for the hearing impaired". Per the definition in WCAG 2, to satisfy this success criterion, whether called captions or subtitles, they would have to provide “synchronized visual and/or text alternative for both speech and non-speech audio information needed to understand the media content” where non-speech information includes “sound effects, music, laughter, speaker identification and location”.

§ *1.2.5 Audio Description (Prerecorded)*

(Level AA)

> Audio description is provided for all prerecorded video content in synchronized media.

§ Applying SC 1.2.5 Audio Description (Prerecorded) to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.2.5.

> NOTE (ADDED)
>
> Audio descriptions (also called "video descriptions", "descriptive narration", and "described videos") describe important visual information needed to understand the video content, including text displayed in the video. Where the main audio track of the video fully describes important visual information, audio descriptions would not be needed at all as the requirement would already be met. When audio descriptions are needed, one way to implement them is by providing a second audio track within the synchronized media.

§ **1.3 Adaptable**

> Create content that can be presented in different ways (for example simpler layout) without losing information or structure.

§ *Applying Guideline 1.3 Adaptable to Non-Web Documents and Software*

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 1.3 applies directly as written.

§ *1.3.1 Info and Relationships*

(Level A)

> Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.

§ Applying SC 1.3.1 Info and Relationships to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.3.1.

> NOTE 1 (ADDED) (FOR NON-WEB DOCUMENTS)
>
> Where non-web documents contain non-standard structure types (roles), it is best practice to map them to a standard structure type as a fall-back solution for the reader.

> NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)
>
> In non-web software, programmatic determinability is best achieved by using the accessibility services of platform software to enable interoperability between the software and assistive technologies and accessibility features of software.

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *1.3.2 Meaningful Sequence*

(Level A)

When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined.

§ Applying SC 1.3.2 Meaningful Sequence to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.3.2.

NOTE (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *1.3.3 Sensory Characteristics*

(Level A)

Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, color, size, visual location, orientation, or sound.

NOTE

For requirements related to color, refer to Guideline 1.4.

§ Applying SC 1.3.3 Sensory Characteristics to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.3.3.

§ *1.3.4 Orientation*

(Level AA)

> Content does not restrict its view and operation to a single display orientation, such as portrait or landscape, unless a specific display orientation is essential.
>
> *NOTE*
>
> *Examples where a particular display orientation may be essential are a bank check, a piano application, slides for a projector or television, or virtual reality content where content is not necessarily restricted to landscape or portrait display orientation.*

§ Applying SC 1.3.4 Orientation to Non-Web Documents

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.3.4, except for non-web documents that will never be displayed on hardware that is reoriented in typical use.

Example (Added) (for non-web documents): Examples of non-web documents that will never be displayed on hardware that is reoriented in typical use include but are not limited to:

- a building directory that is only displayed on displays (of any type including tablets) that are all fixed to the wall in one orientation,
- reports of results of a test that are displayed only on the screen of the testing device, and
- the status report sent to the screen of a copy machine (but not the status report that would be sent to a web interface to the same machine).

§ Applying SC 1.3.4 Orientation to Non-Web Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.3.4, except for non-web software that will never be displayed on hardware that is reoriented in typical use.

NOTE 1 (ADDED) (FOR NON-WEB SOFTWARE)

Non-web software that is only used on hardware that supports a single display orientation, or where it is an application that is displayed only on hardware that is physically fixed in one orientation (e.g. a digital building directory) is excluded by the precondition and therefore does not need to provide support for orientation changes.

Example (Added) (for non-web software): Examples of non-web software that will never be displayed on hardware that is reoriented in typical use include but are not limited to:

- software for a typical calculator that does not support screen re-orientation,
- software menus and controls on a copier or other device that is not intended to be viewed in more than one orientation (but not any remote control software or app for the same device that runs on computers or mobile devices),
- software on a wristwatch that is not intended to be viewed when the watch is not on the wrist, and
- special applications such as software for displays around a building that will only be used on a known set of displays that are all permanently affixed in the same orientation.

NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *1.3.5 Identify Input Purpose*

(Level AA)

The purpose of each input field collecting information about the user can be programmatically determined when:

- The input field serves a purpose identified in the Input Purposes for user interface components section; and
- The content is implemented using technologies with support for identifying the expected meaning for form input data.

Applying SC 1.3.5 Identify Input Purpose to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.3.5.

> NOTE 1 (ADDED)
>
> Non-web software and non-web document technologies that do not provide attributes that support identifying the expected meaning for the form input data are not in scope for this success criterion.

> NOTE 2 (ADDED)
>
> For non-web software and non-web documents that present input fields, the terms for the input purposes would be the equivalent terms to those listed in the WCAG 2 section Input Purposes for User Interface Components that are supported by the technology used.

> NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)
>
> See also the Comments on Closed Functionality.

1.4 Distinguishable

> Make it easier for users to see and hear content including separating foreground from background.

Applying Guideline 1.4 Distinguishable to Non-Web Documents and Software

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 1.4 applies directly as written.

§ *1.4.1 Use of Color*

(Level A)

> Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.
>
> *NOTE*
>
> *This success criterion addresses color perception specifically. Other forms of perception are covered in Guideline 1.3 including programmatic access to color and other visual presentation coding.*

§ Applying SC 1.4.1 Use of Color to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.4.1.

§ *1.4.2 Audio Control*

(Level A)

> If any audio on a web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level.
>
> *NOTE*
>
> *Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the web page (whether or not it is used to meet other success criteria) must meet this success criterion. See Conformance Requirement 5: Non-Interference.*

§ Applying SC 1.4.2 Audio Control to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.4.2, replacing “on a web page” with “in the non-web document or software”, “whole page” with “whole non-web document or software”, and “on the web page” with “in the non-web document or software”; removing “See Conformance Requirement 5: Non-Interference”; and adjusting Note 1 to avoid the use of the normative term "must".

With these substitutions, it would read:

1.4.2 Audio Control: If any audio [in the **non-web document** or **software**] plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level.

> NOTE 1
>
> Since any content that does not meet this success criterion can interfere with a user's ability to use the [whole non-web document or software], [it would be necessary for] all content [in the non-web document or software] (whether or not it is used to meet other success criteria) [to] meet this success criterion.

> NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)
>
> See also the Comments on Closed Functionality.

§ *1.4.3 Contrast (Minimum)*

(Level AA)

> The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following:
>
> **Large Text**
> Large-scale text and images of large-scale text have a contrast ratio of at least 3:1;
>
> **Incidental**
> Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.
>
> **Logotypes**
> Text that is part of a logo or brand name has no contrast requirement.

§ Applying SC 1.4.3 Contrast Minimum to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.4.3.

> NOTE (ADDED) (FOR NON-WEB SOFTWARE)
>
> See also the Comments on Closed Functionality.

§ *1.4.4 Resize Text*

(Level AA)

> Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality.

§ Applying SC 1.4.4 Resize Text to Non-Web Documents

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.4.4.

> NOTE 1 (ADDED)
>
> It is best practice to use only fonts that allow for scaling without loss of quality (e.g. pixelized presentation). This applies in particular to embedded fonts.

> NOTE 2 (ADDED) (FOR NON-WEB DOCUMENTS)
>
> Content for which there are viewers or editors with a 200 percent zoom feature would automatically satisfy this success criterion when used with such viewers or editors, unless the content will not work with that zoom feature.

§ Applying SC 1.4.4 Resize Text to Non-Web Software

This success criterion is problematic to apply directly to non-web software because not all platforms provide text enlargement features that increase all displayed text to 200%. Non-web software needs to work with platform capabilities where they exist, but when the platform has text resizing support up to 200%, but not all text types scale to 200%, it is unreasonable for all apps on a particular platform to be required to build in their own text resizing. Where the platform has text resizing support up to 200%, but where not all text resizes to 200% (because some of the text is already 200% of the default body text size), and provided semantic meaning indicated through differences in text size is maintained, the non-web software should work with the text sizing features to the extent the platform provides. Doing so would still address the user needs identified in the Intent from Understanding Success Criterion 1.4.4. The following criterion is recommended as a substitute for the WCAG language:

Except for captions and images of text, text can be resized without loss of content or functionality and without assistive technology either up to 200 percent or, if the platform provides text resizing capabilities but it does not reach 200 percent for all text, up to the text sizing capabilities of the platform.

NOTE 1 (ADDED)

It is best practice to use only fonts that allow for scaling without loss of quality (e.g. pixelized presentation). This applies in particular to embedded fonts.

NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)

The Intent section in Understanding 1.4.4 Resize Text refers to the ability to allow users to enlarge the text on screen at least up to 200% without needing to use assistive technologies. This means that the non-web software provides some means for enlarging the text 200% (zoom or otherwise) without loss of content or functionality, or that the non-web software works with the platform features to satisfy this success criterion.

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

For non-web software, sometimes the platform provides text scaling to 200% for most, but not all text (e.g. headings, which are naturally large, may not be increased in size to 200%, but other text does increase to 200%). In such cases, authors would only need to support text scaling to the extent provided by user settings in the platform, without losing text-size semantics, content or functionality, to satisfy this success criterion.

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *1.4.5 Images of Text*

(Level AA)

If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following:

Customizable
The image of text can be visually customized to the user's requirements;

Essential
A particular presentation of text is essential to the information being conveyed.

NOTE

Logotypes (text that is part of a logo or brand name) are considered essential.

§ Applying SC 1.4.5 Images of Text to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.4.5.

NOTE (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *1.4.10 Reflow*

(Level AA)

Content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions for:

- Vertical scrolling content at a width equivalent to 320 CSS pixels;
- Horizontal scrolling content at a height equivalent to 256 CSS pixels.

Except for parts of the content which require two-dimensional layout for usage or meaning.

NOTE 1

320 CSS pixels is equivalent to a starting viewport width of 1280 CSS pixels wide at 400% zoom. For web content which is designed to scroll horizontally (e.g., with vertical text), 256 CSS pixels is equivalent to a starting viewport height of 1024 CSS pixels at 400% zoom.

NOTE 2

Examples of content which requires two-dimensional layout are images required for understanding (such as maps and diagrams), video, games, presentations, data tables (not individual cells), and interfaces where it is necessary to keep toolbars in view while manipulating content. It is acceptable to provide two-dimensional scrolling for such parts of the content.

§ Applying SC 1.4.10 Reflow to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.4.10, replacing “web content” with “content”.

With this substitution, it would read:

1.4.10 Reflow: Content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions for:

- Vertical scrolling content at a width equivalent to 320 CSS pixels;
- Horizontal scrolling content at a height equivalent to 256 CSS pixels.

Except for parts of the content which require two-dimensional layout for usage or meaning.

NOTE 1

320 CSS pixels is equivalent to a starting viewport width of 1280 CSS pixels wide at 400% zoom. For [**content**] which is designed to scroll horizontally (e.g., with vertical text), 256 CSS pixels is equivalent to a starting viewport height of 1024 CSS pixels at 400% zoom.

NOTE 2

Examples of content which requires two-dimensional layout are images required for understanding (such as maps and diagrams), video, games, presentations, data tables (not individual cells), and interfaces where it is necessary to keep toolbars in view while manipulating content. It is acceptable to provide two-dimensional scrolling for such parts of the content.

NOTE 3 (ADDED)

In technologies where CSS is not used, the definition of 'CSS pixel' applies as described in Applying “CSS pixel” to Non-Web Documents and Software.

NOTE 4 (ADDED) (FOR NON-WEB DOCUMENTS)

If a non-web document type and its available user agents do not support reflow, it may not be possible for a document of that type to satisfy this success criterion.

NOTE 5 (ADDED) (FOR NON-WEB SOFTWARE)

The intent section refers to the ability for content to reflow (for vertical scrolling content at a width equivalent to 320 CSS pixels, or for horizontal scrolling content at a height equivalent to 256 CSS pixels) when user agent zooming is used to scale content or when the viewport changes in width. For non-web software, this means that when users scale content, adjust the size of a window, dialog, or other resizable content area, or change the screen resolution, the content will reflow without loss of information or functionality, and without requiring scrolling in two dimensions; or that the non-web software works with platform features that satisfy this success criterion.

NOTE 6 (ADDED) (FOR NON-WEB SOFTWARE)

Non-web software will have more frequent cases where two-dimensional layout is relied upon for usage or meaning than what occurs on the Web. For example:

- When the non-web software has a complex user interface with toolbars that need to be visible while manipulating content, as explained in the Intent from Understanding 1.4.10 Reflow.

NOTE 7 (ADDED) (FOR NON-WEB SOFTWARE)

As written, this success criterion can only be met by non-web software where the underlying user agent or platform software can present content at a width equivalent to 320 CSS pixels for vertical scrolling content and a height equivalent to 256 CSS pixels for horizontal scrolling content.

When the underlying user agent or platform software does not support these dimensions for scrolling, reflow is encouraged as this capability is important to people with low vision. As a reasonable benchmark, evaluate at the nearest size to what the Reflow success criterion specifies.

When users modify zoom, scaling, and/or display resolution at the platform software level (e.g. Operating System), it impacts the size of all applications and the platform software itself. This can result in improved readability in some applications but unwanted consequences in others.

NOTE 8 (ADDED) (FOR NON-WEB SOFTWARE)

Some non-web software applications provide a mode of operation where reflow is possible, while other modes are unable to reflow. An example is a document authoring tool, which includes both a "print preview mode" (without reflow, for users to view the spatial formatting) and a "drafting view mode" where reflow is supported. Such software would satisfy this success criterion as long as there is no loss of information or functionality in the drafting view.

NOTE 9 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *1.4.11 Non-text Contrast*

(Level AA)

> The visual presentation of the following have a contrast ratio of at least 3:1 against adjacent color(s):
>
> **User Interface Components**
> Visual information required to identify user interface components and states, except for inactive components or where the appearance of the component is determined by the user agent and not modified by the author;
>
> **Graphical Objects**
> Parts of graphics required to understand the content, except when a particular presentation of graphics is essential to the information being conveyed.

§ Applying SC 1.4.11 Non-text Contrast to Non-Web Documents and Software

This applies directly as written and as described in Intent from Understanding Success Criterion 1.4.11, replacing "user agent" with "user agent or other platform software".

With this substitution, it would read:

1.4.11 Non-text Contrast: The visual presentation of the following have a contrast ratio of at least 3:1 against adjacent color(s):

User Interface Components
Visual information required to identify user interface components and states, except for inactive components or where the appearance of the component is determined by the [**user agent** or other **platform software**] and not modified by the author;

Graphical Objects
Parts of graphics required to understand the content, except when a particular presentation of graphics is essential to the information being conveyed.

NOTE 1 (ADDED)

An example of appearance modification by the author is content that sets the visual style of a control, such as a color or border, to differ from the default style for the user agent or platform.

NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *1.4.12 Text Spacing*

(Level AA)

In content implemented using markup languages that support the following text style properties, no loss of content or functionality occurs by setting all of the following and by changing no other style property:

- Line height (line spacing) to at least 1.5 times the font size;
- Spacing following paragraphs to at least 2 times the font size;
- Letter spacing (tracking) to at least 0.12 times the font size;
- Word spacing to at least 0.16 times the font size.

Exception: Human languages and scripts that do not make use of one or more of these text style properties in written text can conform using only the properties that exist for that combination of language and script.

NOTE 1

Content is not required to use these text spacing values. The requirement is to ensure that when a user overrides the authored text spacing, content or functionality is not lost.

NOTE 2

Writing systems for some languages use different text spacing settings, such as paragraph start indent. Authors are encouraged to follow locally available guidance for improving readability and legibility of text in their writing system.

§ Applying SC 1.4.12 Text Spacing to Non-Web Documents and Software

This applies directly as written and as described in Intent from Understanding Success Criterion 1.4.12.

NOTE 1 (ADDED)

This success criterion only applies to non-web documents and software that are implemented using markup languages and allow the user to modify these text spacing properties.

NOTE 2 (ADDED) (FOR NON-WEB DOCUMENTS)

There are several mechanisms that allow users to modify a document's text spacing properties of content implemented in markup languages. For example, an eBook technology may have an available user agent that allows users to override document text styles. When such a mechanism is available, the success criterion requires that the content responds appropriately to it.

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

There are several mechanisms that allow users to modify software's text spacing properties of content implemented in markup languages. For example, a software application may provide a "user style sheet" facility to modify the appearance of the software's own user interface. This success criterion does not mean that non-web software needs to implement their own mechanisms to allow users to set text spacing; however, when such a mechanism is available, the success criterion requires that the content responds appropriately to it.

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

"Content implemented using markup languages" includes parts of software that use markup internally to define a user interface. Examples of markup languages that are used internally to define a software user interface include but are not limited to: HTML (e.g., in Electron applications or iOS application web views), XAML, XML (e.g., in Android application layouts), and XUL.

NOTE 5 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *1.4.13 Content on Hover or Focus*

(Level AA)

Where receiving and then removing pointer hover or keyboard focus triggers additional content to become visible and then hidden, the following are true:

Dismissible
: A mechanism is available to dismiss the additional content without moving pointer hover or keyboard focus, unless the additional content communicates an input error or does not obscure or replace other content;

Hoverable
: If pointer hover can trigger the additional content, then the pointer can be moved over the additional content without the additional content disappearing;

Persistent
: The additional content remains visible until the hover or focus trigger is removed, the user dismisses it, or its information is no longer valid.

Exception: The visual presentation of the additional content is controlled by the user agent and is not modified by the author.

NOTE 1

Examples of additional content controlled by the user agent include browser tooltips created through use of the HTML `title` attribute [HTML].

NOTE 2

Custom tooltips, sub-menus, and other nonmodal popups that display on hover and focus are examples of additional content covered by this criterion.

NOTE 3

This criterion applies to content that appears in addition to the triggering component itself. Since hidden components that are made visible on keyboard focus (such as links used to skip to another part of a page) do not present additional content they are not covered by this criterion.

§ Applying SC 1.4.13 Content on Hover or Focus to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 1.4.13, replacing "user agent" with "user agent or other platform software", "browser tooltips" with "tooltips", "the HTML title attribute" with "user interface object attributes", "links" with "links or other UI controls that behave like a link", and "a page" with "the non-web document or software".

With these substitutions, it would read:

1.4.13 Content on Hover or Focus: Where receiving and then removing pointer hover or keyboard focus triggers additional content to become visible and then hidden, the following are true:

Dismissible
: A mechanism is available to dismiss the additional content without moving pointer hover or keyboard focus, unless the additional content communicates an input error or does not obscure or replace other content;

Hoverable
: If pointer hover can trigger the additional content, then the pointer can be moved over the additional content without the additional content disappearing;

Persistent
: The additional content remains visible until the hover or focus trigger is removed, the user dismisses it, or its information is no longer valid.

Exception: The visual presentation of the additional content is controlled by the [**user agent** or other **platform software**] and is not modified by the author.

> NOTE 1
>
> Examples of additional content controlled by the [user agent or other platform software] include [tooltips] created through use of [user interface object attributes].

> NOTE 2
> Custom tooltips, sub-menus, and other nonmodal popups that display on hover and focus are examples of additional content covered by this criterion.

NOTE 3

This criterion applies to content that appears in addition to the triggering component itself. Since hidden components that are made visible on keyboard focus (such as [links or other UI controls that behave like a link] used to skip to another part of [the non-web document or software]) do not present additional content they are not covered by this criterion.

2. Operable

> User interface components and navigation must be operable.

Applying Principle 2 Operable to Non-Web Documents and Software

In WCAG 2, the Principles are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Principle 2 applies directly as written.

2.1 Keyboard Accessible

> Make all functionality available from a keyboard.

Applying Guideline 2.1 Keyboard Accessible to Non-Web Documents and Software

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 2.1 applies directly as written.

§ *2.1.1 Keyboard*

(Level A)

> All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints.
>
> *NOTE 1*
>
> *This exception relates to the underlying function, not the input technique. For example, if using handwriting to enter text, the input technique (handwriting) requires path-dependent input but the underlying function (text input) does not.*
>
> *NOTE 2*
>
> *This does not forbid and should not discourage providing mouse input or other input methods in addition to keyboard operation.*

§ Applying SC 2.1.1 Keyboard to Non-Web Documents

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.1.1.

§ Applying SC 2.1.1 Keyboard to Non-Web Software

Where ICT is or includes non-web software that can be run on a software platform that provides a device-independent keyboard interface service, this applies directly as written, and as described in Intent from Understanding Success Criterion 2.1.1.

NOTE 1 (ADDED) (FOR NON-WEB SOFTWARE)

Keyboard interface does not refer to a physical device but to the service of platform software (e.g. operating system, browser, etc.) that provides the software with keystrokes from any keyboard or keyboard substitute. When the non-web software supports such a device-independent service of the platform software, and the non-web software functionality is made fully operable through the service, then this success criterion would be satisfied.

NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)

A "device-independent keyboard interface service" refers to the platform service that provides keystrokes to any software running on the platform.

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

Inclusion of an on-screen keyboard can be done as well but does not satisfy this requirement since it does not allow for the use of keyboard alternatives whereas support of input from the device-independent keyboard interface service does.

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

This success criterion does not imply that non-web software always needs to directly support a keyboard or “keyboard interface” if one is not provided by the platform software. But if one is provided, the software needs to make all functionality available through it - unless the exception applies.

NOTE 5 (ADDED) (FOR NON-WEB SOFTWARE)

This success criterion also does not imply that non-web software always needs to provide its own virtual keyboard. But if it does, then the non-web software still needs to support keyboard input from any keyboard interface provided by the platform software.

NOTE 6 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *2.1.2 No Keyboard Trap*

(Level A)

If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away.

NOTE

Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the web page (whether it is used to meet other success criteria or not) must meet this success criterion. See Conformance Requirement 5: Non-Interference.

§ Applying SC 2.1.2 No Keyboard Trap to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.1.2, replacing “page” with “non-web document or software” and “on the web page” with "in the non-web document or software"; removing “See Conformance Requirement 5: Non-Interference”; and adjusting Note 1 to avoid the use of the normative term "must".

With these substitutions, it would read:

2.1.2 No Keyboard Trap: If keyboard focus can be moved to a component of the **[non-web document** or **software]** using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away.

NOTE 1

Since any content that does not meet this success criterion can interfere with a user's ability to use the whole [**non-web document** or **software**], [it would be necessary for] all content [in the non-web document or software] (whether it is used to meet other success criteria or not) [to] meet this success criterion.

NOTE 2 (ADDED)

Standard exit methods may vary by platform. For example, on many desktop platforms, the Escape key is a standard method for exiting.

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

This criterion applies when focus can be moved using a keyboard interface. Some software may accept input from a keyboard, keypad, or controller, yet not offer any mechanism for focus; for example, the keys are mapped directly to functions without moving focus between on-screen controls. In this case, there is no concept of focus, and therefore keyboard traps cannot exist and this success criterion would be satisfied.

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *2.1.4 Character Key Shortcuts*

(Level A)

> If a keyboard shortcut is implemented in content using only letter (including upper- and lower-case letters), punctuation, number, or symbol characters, then at least one of the following is true:
>
> **Turn off**
> A mechanism is available to turn the shortcut off;
>
> **Remap**
> A mechanism is available to remap the shortcut to include one or more non-printable keyboard keys (e.g., Ctrl, Alt);
>
> **Active only on focus**
> The keyboard shortcut for a user interface component is only active when that component has focus.

§ Applying SC 2.1.4 Character Key Shortcuts to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.1.4.

> NOTE 1 (ADDED) (FOR NON-WEB SOFTWARE)
>
> The WCAG2ICT interpretation is that a long press of a key (2 seconds or more) and other accessibility features provided by the platform do not meet the WCAG definition of a keyboard shortcut. See the keyboard shortcut definition for more details.

> NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)
>
> See also the Comments on Closed Functionality.

§ **2.2 Enough Time**

> Provide users enough time to read and use content.

§ *Applying Guideline 2.2 Enough Time to Non-Web Documents and Software*

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 2.2 applies directly as written.

§ *2.2.1 Timing Adjustable*

(Level A)

For each time limit that is set by the content, at least one of the following is true:

Turn off
The user is allowed to turn off the time limit before encountering it; or

Adjust
The user is allowed to adjust the time limit before encountering it over a wide range that is at least ten times the length of the default setting; or

Extend
The user is warned before time expires and given at least 20 seconds to extend the time limit with a simple action (for example, "press the space bar"), and the user is allowed to extend the time limit at least ten times; or

Real-time Exception
The time limit is a required part of a real-time event (for example, an auction), and no alternative to the time limit is possible; or

Essential Exception
The time limit is essential and extending it would invalidate the activity; or

20 Hour Exception
The time limit is longer than 20 hours.

NOTE

This success criterion helps ensure that users can complete tasks without unexpected changes in content or context that are a result of a time limit. This success criterion should be considered in conjunction with Success Criterion 3.2.1, which puts limits on changes of content or context as a result of user action.

§ Applying SC 2.2.1 Timing Adjustable to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.2.1, replacing " content" with "non-web document or software".

With this substitution, it would read:

2.2.1 Timing Adjustable: For each time limit that is set by the **[non-web document** or **software]**, at least one of the following is true:

Turn off
The user is allowed to turn off the time limit before encountering it; or

Adjust
The user is allowed to adjust the time limit before encountering it over a wide range that is at least ten times the length of the default setting; or

Extend
The user is warned before time expires and given at least 20 seconds to extend the time limit with a simple action (for example, "press the space bar"), and the user is allowed to extend the time limit at least ten times; or

Real-time Exception
The time limit is a required part of a real-time event (for example, an auction), and no alternative to the time limit is possible; or

Essential Exception
The time limit is essential and extending it would invalidate the activity; or

20 Hour Exception
The time limit is longer than 20 hours.

> NOTE
>
> This success criterion helps ensure that users can complete tasks without unexpected changes in content or context that are a result of a time limit. This success criterion should be considered in conjunction with Success Criterion 3.2.1, which puts limits on changes of content or context as a result of user action.

§ *2.2.2 Pause, Stop, Hide*

(Level A)

For moving, blinking, scrolling, or auto-updating information, all of the following are true:

Moving, blinking, scrolling

For any moving, blinking or scrolling information that (1) starts automatically, (2) lasts more than five seconds, and (3) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it unless the movement, blinking, or scrolling is part of an activity where it is essential; and

Auto-updating

For any auto-updating information that (1) starts automatically and (2) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it or to control the frequency of the update unless the auto-updating is part of an activity where it is essential.

NOTE 1

For requirements related to flickering or flashing content, refer to Guideline 2.3.

NOTE 2

Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the web page (whether it is used to meet other success criteria or not) must meet this success criterion. See Conformance Requirement 5: Non-Interference.

NOTE 3

Content that is updated periodically by software or that is streamed to the user agent is not required to preserve or present information that is generated or received between the initiation of the pause and resuming presentation, as this may not be technically possible, and in many situations could be misleading to do so.

NOTE 4

An animation that occurs as part of a preload phase or similar situation can be considered essential if interaction cannot occur during that phase for all users and if not indicating progress could confuse users or cause them to think that content was frozen or broken.

§ Applying SC 2.2.2 Pause, Stop, Hide to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.2.2, replacing “page” with “non-web document or software” and “on the web page” with “in the non-web document or software”; removing “See Conformance Requirement 5: Non-Interference” in Note 2 of the success criterion; and adjusting Note 2 to avoid the use of the normative term "must".

With these substitutions, it would read:

2.2.2 Pause, Stop, Hide: For moving, blinking, scrolling, or auto-updating information, all of the following are true:

Moving, blinking, scrolling
: For any moving, blinking or scrolling information that (1) starts automatically, (2) lasts more than five seconds, and (3) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it unless the movement, blinking, or scrolling is part of an activity where it is essential; and

Auto-updating
: For any auto-updating information that (1) starts automatically and (2) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it or to control the frequency of the update unless the auto-updating is part of an activity where it is essential.

> NOTE 1
>
> For requirements related to flickering or flashing content, refer to Guideline 2.3.

> NOTE 2
>
> Since any content that does not meet this success criterion can interfere with a user's ability to use the whole [**non-web document** or **software**], [it would be necessary for] all content [in the non-web document or software] (whether it is used to meet other success criteria or not) [to] meet this success criterion.

NOTE 3

Content that is updated periodically by software or that is streamed to the user agent is not required to preserve or present information that is generated or received between the initiation of the pause and resuming presentation, as this may not be technically possible, and in many situations could be misleading to do so.

NOTE 4

An animation that occurs as part of a preload phase or similar situation can be considered essential if interaction cannot occur during that phase for all users and if not indicating progress could confuse users or cause them to think that content was frozen or broken.

NOTE 5 (ADDED)

While the success criterion uses the term “information”, the WCAG 2 Intent from Understanding Success Criterion 2.2.2 makes it clear that this is to be applied to all content. Any content, even if just decorative, that is updated automatically, blinks, or moves may create an accessibility barrier.

§ 2.3 Seizures and Physical Reactions

> Do not design content in a way that is known to cause seizures or physical reactions.

§ *Applying Guideline 2.3 Seizures and Physical Reactions to Non-Web Documents and Software*

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 2.3 applies directly as written.

§ *2.3.1 Three Flashes or Below Threshold*

(Level A)

> Web pages do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds.
>
> *NOTE*
>
> *Since any content that does not meet this success criterion can interfere with a user's ability to use the whole page, all content on the web page (whether it is used to meet other success criteria or not) must meet this success criterion. See Conformance Requirement 5: Non-Interference.*

§ Applying SC 2.3.1 Three Flashes or Below Threshold to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.3.1, replacing “web pages” with “non-web documents or software” , “page” with “non-web document or software”, and “on the web page” with “in the non-web document or software”; removing “See Conformance Requirement 5: Non-Interference”; and adjusting Note 1 to avoid the use of the normative term "must".

With these substitutions, it would read:

2.3.1 Three Flashes or Below Threshold: [**Non-web documents** or **software**] do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds.

NOTE 1

Since any content that does not meet this success criterion can interfere with a user's ability to use the whole [**non-web document** or **software**], [it would be necessary for] all content [in the non-web document or software] (whether it is used to meet other success criteria or not) [to] meet this success criterion.

NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)

This requirement applies to flashing of content on a screen and flashing of any other type caused by the ICT.

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

This requirement applies to those visual elements produced by the ICT itself. Content from an external source that is presented through the ICT, is the responsibility of the source. The requirement does not require the ICT to examine or modify such externally supplied content in any way.

Example (Added) (for non-web software): Examples of ICT that presents content from an external source include TVs playing broadcast programs and media players that are playing content provided by the user.

§ 2.4 Navigable

Provide ways to help users navigate, find content, and determine where they are.

§ *Applying Guideline 2.4 Navigable to Non-Web Documents and Software*

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 2.4 applies directly as written.

§ *2.4.1 Bypass Blocks*

(Level A)

> A mechanism is available to bypass blocks of content that are repeated on multiple web pages.

§ Applying SC 2.4.1 Bypass Blocks to Non-Web Documents and Software

This applies directly as written and described in Intent from Understanding Success Criterion 2.4.1, replacing “on multiple web pages” with “in multiple non-web documents in a set of non-web documents, or in multiple software programs in a set of software programs” to explicitly state that the multiple documents (or software programs) are part of a set rather than any two documents or pieces of software.

With these substitutions, this success criterion would read:

2.4.1 Bypass Blocks: A mechanism is available to bypass blocks of content that are repeated [in multiple **non-web documents** in a **set of non-web documents**, or in multiple **software programs** in a **set of software programs**].

NOTE 1 (ADDED)

See set of documents and set of software programs in the Key Terms section to determine when a group of documents or software programs is considered a set for this success criterion. Those implementing this document (WCAG2ICT) will need to consider if this success criterion is appropriate to apply to non-web documents and software. See the Interpretation of Web Terminology in a Non-web Context.

NOTE 2 (ADDED)

Individual documents or software programs (not in a set) would automatically satisfy this success criterion because this success criterion applies only to things that appear in a set.

NOTE 3 (ADDED)

Although not required by the success criterion, being able to bypass blocks of content that are repeated *within* non-web documents or software directly addresses user needs identified in the Intent section for this success criterion, and is generally considered best practice.

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

Sets of software that meet this definition appear to be extremely rare.

NOTE 5 (ADDED) (FOR NON-WEB SOFTWARE)

Many software user interface components have built-in mechanisms to navigate directly to / among them, which also have the effect of skipping over or bypassing blocks of content.

NOTE 6 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *2.4.2 Page Titled*

(Level A)

> Web pages have titles that describe topic or purpose.

§ Applying SC 2.4.2 Page Titled to Non-Web Documents

This success criterion is problematic to apply directly to non-web documents through simple word substitution because not all document formats provide support for a programmatically determinable Title property, and document titles don't always describe the topic or purpose of the document. File names, as the WCAG 2 Understanding document specifies, also rarely describe the topic or purpose of the document – especially where the document names are not under the author's control. However, where the document authoring tool or technology provides the capability to supply a title or name for a document, when the non-web document utilizes the Title property to provide a unique title or name inside of each document, and/or when a meaningful file name can be supplied, the user can more easily find it or understand its purpose. This would address the user needs identified in the Intent from Understanding Success Criterion 2.4.2. The following criterion is recommended as a substitute for the WCAG language:

2.4.2 Non-web Document Titled: In non-web documents implemented in a format that supports a programmatically determinable Title property that is editable using common authoring tools for that document format, the non-web document has a title that describes the name, topic, or purpose.

> NOTE (ADDED) (FOR NON-WEB DOCUMENTS)
>
> The Title property is specified as "editable through that document format's common authoring tools" so that authors can view and edit the Title without requiring specialized or external metadata utilities. "Common authoring tools" are the most readily available tools used for editing a particular document type.

§ Applying SC 2.4.2 Page Titled to Non-Web Software

This success criterion is problematic to apply directly to non-web software through simple word substitution because application titles rarely describe the topic or purpose of the software. However, where the platform supports a programmatic title or name for a software window or screen, when a software application utilizes that feature to provide a unique title or name for each window or screen, the user can more easily find it or understand its purpose. This would address the user needs identified in the Intent from Understanding Success Criterion 2.4.2. The following criterion is recommended as a substitute for the WCAG language:

2.4.2 Non-web Software Titled: In non-web software implemented on a platform that supports a Title property for windows or screens, the non-web software provides titles that describe the name, topic or purpose of each window or screen.

§ *2.4.3 Focus Order*

(Level A)

> If a web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.

§ Applying SC 2.4.3 Focus Order to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.4.3 replacing “a web page” with “non-web documents or software”.

With this substitution, it would read:

2.4.3 Focus Order: If **[non-web documents or software]** can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.

§ *2.4.4 Link Purpose (In Context)*

(Level A)

> The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general.

§ Applying SC 2.4.4 Link Purpose (In Context) to Non-Web Documents and Software

This applies directly as written and as described in Intent from Understanding Success Criterion 2.4.4.

> NOTE 1 (ADDED)
>
> In non-web documents or software, a “link” is any user interface control that behaves like a hypertext link.

> NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)
>
> See also the Comments on Closed Functionality.

§ *2.4.5 Multiple Ways*

(Level AA)

> More than one way is available to locate a web page within a set of web pages except where the web page is the result of, or a step in, a process.

§ Applying SC 2.4.5 Multiple Ways to Non-Web Documents and Software

This applies directly as written and described in Intent from Understanding Success Criterion 2.4.5, replacing “web page within a set of web pages” with “non-web document within a set of non-web documents, or a set of software programs within a set of software programs” and "the web page" with "the non-web document or software program".

With these substitutions, this success criterion would read:

2.4.5 Multiple Ways: More than one way is available to locate a [**non-web document** within a **set of non-web documents**, or a **software program** within a **set of software programs**] except where [the non-web document or software program] is the result of, or a step in, a process.

NOTE 1 (ADDED)

See set of documents and set of software programs in the Key Terms section to determine when a group of documents or software programs is considered a set for this success criterion. Those implementing this document (WCAG2ICT) will need to consider if this success criterion is appropriate to apply to non-web documents and software. See the Interpretation of Web Terminology in a Non-web Context.

NOTE 2 (ADDED)

The definitions of “set of documents” and “set of software programs” in the Key Terms section are predicated on the ability to navigate from each element of the set to each other, and navigation is a type of locating. So the mechanism used to navigate between elements of the set will be one way of locating information in the set. Non-web environments, generally major operating systems with browse and search capabilities, often provide infrastructure and tools that provide mechanisms for locating content in a set of non-web documents or a set of software programs. For example, it may be possible to browse through the files or programs that make up a set, or search within members of the set for the names of other members. A file directory would be the equivalent of a site map for documents in a set, and a search function in a file system would be equivalent to a web search function for web pages. Such facilities may provide additional ways of locating information in the set.

NOTE 3 (ADDED)

While some users may find it useful to have multiple ways to locate some groups of user interface elements within a non-web document or software program, this is not required by the success criterion (and may pose difficulties in some situations).

NOTE 4 (ADDED)

The definitions of "set of documents" and "set of software programs" in WCAG2ICT require every item in the set to be independently reachable, and so nothing in such a set can be a "step in a process" that can't be reached any other way. The purpose of the exception—that items in a process are exempt from satisfying this success criterion—is achieved by the definition of set.

NOTE 5 (ADDED) (FOR NON-WEB SOFTWARE)

Sets of software that meet this definition appear to be extremely rare.

NOTE 6 (ADDED) (FOR NON-WEB SOFTWARE)

An example of the use of "a software program that is part of process", that would meet the exception for this success criterion, would be one where programs are interlinked but the interlinking depends on program A being used before program B, for validation or to initialize the dataset, etc.

NOTE 7 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *2.4.6 Headings and Labels*

(Level AA)

> Headings and labels describe topic or purpose.

§ Applying SC 2.4.6 Headings and Labels to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.4.6.

> NOTE (ADDED) (FOR NON-WEB SOFTWARE)
>
> In non-web software, headings and labels are used to describe sections of content and controls respectively. In some cases it may be unclear whether a piece of static text is a heading or a label. But whether treated as a label or a heading, the requirement is the same: that if they are present they describe the topic or purpose of the item(s) they are associated with.

§ *2.4.7 Focus Visible*

(Level AA)

> Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible.

§ Applying SC 2.4.7 Focus Visible to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.4.7.

> NOTE (ADDED) (FOR NON-WEB SOFTWARE)
>
> See also the Comments on Closed Functionality.

§ *2.4.11 Focus Not Obscured (Minimum)*

(Level AA)

> When a user interface component receives keyboard focus, the component is not entirely hidden due to author-created content.
>
> > *NOTE 1*
> >
> > *Where content in a configurable interface can be repositioned by the user, then only the initial positions of user-movable content are considered for testing and conformance of this success criterion.*
>
> > *NOTE 2*
> >
> > *Content opened by the* user *may obscure the component receiving focus. If the user can reveal the focused component without advancing the keyboard focus, the component with focus is not considered visually hidden due to author-created content.*

§ Applying SC 2.4.11 Focus not Obscured to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.4.11.

§ **2.5 Input Modalities**

> Make it easier for users to operate functionality through various inputs beyond keyboard.

§ *Applying Guideline 2.5 Input Modalities to Non-Web Documents and Software*

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 2.5 applies directly as written.

§ *2.5.1 Pointer Gestures*

(Level A)

> All functionality that uses multipoint or path-based gestures for operation can be operated with a single pointer without a path-based gesture, unless a multipoint or path-based gesture is essential.
>
> > *NOTE*
> >
> > *This requirement applies to web content that interprets pointer actions (i.e., this does not apply to actions that are required to operate the user agent or assistive technology).*

§ Applying SC 2.5.1 Pointer Gestures to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.5.1, making changes to the notes for non-web documents by replacing “web content” with "content", for non-web software by replacing "web content that interprets" with "non-web software that interprets" and "user agent" with "underlying platform software".

With these substitutions, the notes would read:

> NOTE 1 (FOR NON-WEB DOCUMENTS)
>
> This requirement applies to [**content**] that interprets pointer actions (i.e., this does not apply to actions that are required to operate the user agent or assistive technology).

NOTE 2 (ADDED) (FOR NON-WEB DOCUMENTS)

Multipoint and path-based gestures are less common in non-web documents. An example where a non-web document author could add such gestures is an interactive prototype document created in a software design tool.

NOTE 3 (FOR NON-WEB SOFTWARE)

This requirement applies to [non-web **software** that interprets] pointer actions (i.e., this does not apply to actions that are required to operate the [underlying **platform software**] or assistive technology).

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

This requirement also applies to platform software, such as user agents, assistive technology software, and operating systems. Each layer is responsible for its own pointer actions only, not for those in an underlying layer.

§ *2.5.2 Pointer Cancellation*

(Level A)

For functionality that can be operated using a single pointer, at least one of the following is true:

No Down-Event
The down-event of the pointer is not used to execute any part of the function;

Abort or Undo
Completion of the function is on the up-event, and a mechanism is available to abort the function before completion or to undo the function after completion;

Up Reversal
The up-event reverses any outcome of the preceding down-event;

Essential
Completing the function on the down-event is essential.

NOTE 1

Functions that emulate a keyboard or numeric keypad key press are considered essential.

NOTE 2

This requirement applies to web content that interprets pointer actions (i.e., this does not apply to actions that are required to operate the user agent or assistive technology).

§ Applying SC 2.5.2 Pointer Cancellation to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.5.2, making changes to the notes for non-web documents by replacing “web content” with "content", for non-web software by replacing "web content that interprets" with "non-web software that interprets" and "user agent" with "underlying platform software".

With these substitutions, the notes would read:

NOTE 1 (FOR NON-WEB DOCUMENTS)

Functions that emulate a keyboard or numeric keypad key press are considered essential.

NOTE 2 (FOR NON-WEB DOCUMENTS)

This requirement applies to [**content**] that interprets pointer actions (i.e., this does not apply to actions that are required to operate the user agent or assistive technology).

NOTE 3 (ADDED) (FOR NON-WEB DOCUMENTS)

Content that interprets pointer actions and controls which events are used for executing functionality is less common in non-web documents. An example where a non-web document author could add such functionality is an interactive prototype document created in a software design tool.

NOTE 4 (FOR NON-WEB SOFTWARE)

Functions that emulate a keyboard or numeric keypad key press are considered essential.

Example (Added) (for non-web software): Examples of essential functionality for non-web software are features for meeting environmental energy usage requirements (like waking a device from sleep, power saver mode, and low power state).

NOTE 5 (FOR NON-WEB SOFTWARE)

This requirement applies to [non-web **software** that interprets] pointer actions (i.e., this does not apply to actions that are required to operate the [underlying **platform software**] or assistive technology).

NOTE 6 (ADDED) (FOR NON-WEB SOFTWARE)

This requirement also applies to platform software, such as user agents, assistive technology software, and operating systems. Each layer is responsible for its own pointer actions only, not for those in an underlying layer.

NOTE 7 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *2.5.3 Label in Name*

(Level A)

For user interface components with labels that include text or images of text, the name contains the text that is presented visually.

NOTE

A best practice is to have the text of the label at the start of the name.

§ Applying SC 2.5.3 Label in Name to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.5.3.

NOTE (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *2.5.4 Motion Actuation*

(Level A)

> Functionality that can be operated by device motion or user motion can also be operated by user interface components and responding to the motion can be disabled to prevent accidental actuation, except when:
>
> **Supported Interface**
> The motion is used to operate functionality through an accessibility supported interface;
>
> **Essential**
> The motion is essential for the function and doing so would invalidate the activity.

§ Applying SC 2.5.4 Motion Actuation to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.5.4.

§ *2.5.7 Dragging Movements*

(Level AA)

> All functionality that uses a dragging movement for operation can be achieved by a single pointer without dragging, unless dragging is essential or the functionality is determined by the user agent and not modified by the author.
>
> *NOTE*
>
> *This requirement applies to web content that interprets pointer actions (i.e., this does not apply to actions that are required to operate the user agent or assistive technology).*

§ Applying SC 2.5.7 Dragging Movements to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.5.7, replacing "user agent" with "user agent or other platform software" and by making changes to the

notes for non-web documents by replacing “web content” with "content", and for non-web software by replacing "web content that interprets" with "non-web software that interprets" and "user agent" with "underlying platform software".

With these substitutions, it would read:

2.5.7 Dragging Movements: All functionality that uses a dragging movement for operation can be achieved by a single pointer without dragging, unless dragging is essential or the functionality is determined by the [**user agent** or other **platform software**] and not modified by the author.

NOTE 1 (FOR NON-WEB DOCUMENTS)

This requirement applies to [**content**] that interprets pointer actions (i.e., this does not apply to actions that are required to operate the user agent or assistive technology).

NOTE 2 (ADDED) (FOR NON-WEB DOCUMENTS)

Dragging movements for operation are less common in non-web documents. An example where a document author could add dragging functionality is an interactive prototype document created in a software design tool.

NOTE 3 (FOR NON-WEB SOFTWARE)

This requirement applies to [non-web **software** that interprets] pointer actions (i.e., this does not apply to actions that are required to operate the [underlying platform software] or assistive technology).

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

This requirement also applies to platform software, such as user agents, assistive technology software, and operating systems. Each layer is responsible for its own pointer actions only, not for those in an underlying layer.

§ *2.5.8 Target Size (Minimum)*

(Level AA)

The size of the target for pointer inputs is at least 24 by 24 CSS pixels, except when:

Spacing
Undersized targets (those less than 24 by 24 CSS pixels) are positioned so that if a 24 CSS pixel diameter circle is centered on the bounding box of each, the circles do not intersect another target or the circle for another undersized target;

Equivalent
The function can be achieved through a different control on the same page that meets this criterion;

Inline
The target is in a sentence or its size is otherwise constrained by the line-height of non-target text;

User Agent Control
The size of the target is determined by the user agent and is not modified by the author;

Essential
A particular presentation of the target is essential or is legally required for the information being conveyed.

NOTE 1

Targets that allow for values to be selected spatially based on position within the target are considered one target for the purpose of the success criterion. Examples include sliders, color pickers displaying a gradient of colors, or editable areas where you position the cursor.

NOTE 2

For inline targets the line-height should be interpreted as perpendicular to the flow of text. For example, in a language displayed vertically, the line-height would be horizontal.

§ Applying SC 2.5.8 Target Size (Minimum) to Non-Web Documents and Software:

This applies directly as written, and as described in Intent from Understanding Success Criterion 2.5.8, replacing "user agent" with "user agent or other platform software" and "on the same page" with "in the same non-web document or software".

With these substitutions, it would read:

2.5.8 Target Size (Minimum): The size of the target for pointer inputs is at least 24 by 24 CSS pixels, except when:

Spacing
: Undersized targets (those less than 24 by 24 CSS pixels) are positioned so that if a 24 CSS pixel diameter circle is centered on the bounding box of each, the circles do not intersect another target or the circle for another undersized target;

Equivalent
: The function can be achieved through a different control [in the same **non-web document** or **software**] that meets this criterion.

Inline
: The target is in a sentence or its size is otherwise constrained by the line-height of non-target text;

[User agent or other platform software] control
: The size of the target is determined by the [**user agent** or other **platform software**] and is not modified by the author;

Essential
: A particular presentation of the target is essential or is legally required for the information being conveyed.

> NOTE 1
>
> Targets that allow for values to be selected spatially based on position within the target are considered one target for the purpose of the success criterion. Examples include sliders, color pickers displaying a gradient of colors, or editable areas where you position the cursor.

NOTE 2

For inline targets the line-height should be interpreted as perpendicular to the flow of text. For example, in a language displayed vertically, the line-height would be horizontal.

NOTE 3 (ADDED)

In technologies where CSS is not used, the definition of 'CSS pixel' applies as described in Applying “CSS pixel” to Non-Web Documents and Software.

NOTE 4 (ADDED) (FOR NON-WEB DOCUMENTS)

Some non-web document formats are designed for viewing at a wide range of zoom levels provided by the user agent. However, the commonly available user agents for these formats may lack a consistent base zoom level from which to evaluate this criterion. For such documents, evaluate target sizes at a zoom level that aligns with the intended usage of the content.

NOTE 5 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

3. Understandable

> Information and the operation of the user interface must be understandable.

Applying Principle 3 Understandable to Non-Web Documents and Software

In WCAG 2, the Principles are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Principle 3 applies directly as written.

3.1 Readable

> Make text content readable and understandable.

Applying Guideline 3.1 Readable to Non-Web Documents and Software

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 3.1 applies directly as written.

3.1.1 Language of Page

(Level A)

> The default human language of each web page can be programmatically determined.

Applying SC 3.1.1 Language of Page to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.1.1 replacing “each web page” with “non-web documents or software”.

With this substitution, it would read:

3.1.1 Language of Page: The default human language of [**non-web documents** or **software**] can be programmatically determined.

NOTE 1 (ADDED) (FOR NON-WEB SOFTWARE)

Where software platforms provide a “locale / language” setting, applications that use that setting and render their interface in that “locale / language” would satisfy this success criterion. Applications that do not use the platform “locale / language” setting but instead use an accessibility-supported method for exposing the human language of the non-web software would also satisfy this success criterion. Applications implemented in technologies where assistive technologies cannot determine the human language and that do not support the platform “locale / language” setting may not be able to satisfy this success criterion in that locale / language.

NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *3.1.2 Language of Parts*

(Level AA)

> The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text.

§ Applying SC 3.1.2 Language of Parts to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.1.2 replacing “content” with “non-web document or software”.

With this substitution, it would read:

3.1.2 Language of Parts: The human language of each passage or phrase in the [**non-web document or software**] can be programmatically determined except for proper names, technical terms, words of

indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text.

> NOTE 1 (ADDED)
>
> Examples of programmatic identification include language metadata or markup. There are some non-web software and non-web document technologies where there is no assistive technology supported method for marking the language for the different passages or phrases in the non-web document or software, and it would not be possible to satisfy this success criterion with those technologies.

> NOTE 2 (ADDED) (FOR NON-WEB DOCUMENTS)
>
> Inheritance is one common method. For example, where the primary language of a non-web document is programmatically determinable, it can be assumed that all of the text or user interface elements within that document will be using the same language unless it is indicated.

> NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)
>
> See also the Comments on Closed Functionality.

§ 3.2 Predictable

> Make web pages appear and operate in predictable ways.

§ *Applying Guideline 3.2 Predictable to Non-Web Documents and Software*

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 3.2 applies directly as written, replacing “web pages” with “non-web documents or software”.

With this substitution, it would read:

Guideline 3.2 Predictable: Make [**non-web documents** or **software**] appear and operate in predictable ways.

§ *3.2.1 On Focus*

(Level A)

> When any user interface component receives focus, it does not initiate a change of context.

§ Applying SC 3.2.1 On Focus to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.2.1.

> NOTE (ADDED)
>
> Some compound documents and their user agents are designed to provide significantly different viewing and editing functionality depending upon what portion of the compound document is being interacted with (e.g. a presentation that contains an embedded spreadsheet, where the menus and toolbars of the user agent change depending upon whether the user is interacting with the presentation content, or the embedded spreadsheet content). If the user uses a mechanism other than putting focus on that portion of the compound document with which they mean to interact (e.g. by a menu choice or special keyboard gesture), any resulting change of context wouldn't be subject to this success criterion because it was not caused by a change of focus.

§ *3.2.2 On Input*

(Level A)

> Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component.

§ Applying SC 3.2.2 On Input to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.2.2.

§ *3.2.3 Consistent Navigation*

(Level AA)

> Navigational mechanisms that are repeated on multiple web pages within a set of web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user.

§ Applying SC 3.2.3 Consistent Navigation to Non-Web Documents and Software

This applies directly as written and described in Intent from Understanding Success Criterion 3.2.3, replacing "on multiple web pages within a set of web pages" with "in multiple non-web documents within a set of non-web documents, or in multiple non-web software programs within a set of software programs".

With these substitutions, it would read:

3.2.3 Consistent Navigation: Navigational mechanisms that are repeated [in multiple **non-web documents** within a **set of non-web documents**, or in multiple **software programs** within a **set of software programs**] occur in the same relative order each time they are repeated, unless a change is initiated by the user.

NOTE 1 (ADDED)

See set of documents and set of software programs in the Key Terms section to determine when a group of documents or software programs is considered a set for this success criterion. Those implementing this document (WCAG2ICT) will need to consider if this success criterion is appropriate to apply to non-web documents and software. See the Interpretation of Web Terminology in a Non-web Context.

NOTE 2 (ADDED)

Although not required by this success criterion, ensuring that navigation elements have consistent order when repeated within non-web documents or software programs directly addresses user needs identified in the Intent section for this success criterion, and is generally considered best practice.

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

Sets of software that meet this definition appear to be extremely rare.

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *3.2.4 Consistent Identification*

(Level AA)

> Components that have the same functionality within a set of web pages are identified consistently.

§ Applying SC 3.2.4 Consistent Identification to Non-Web Documents and Software

This applies directly as written and described in Intent from Understanding Success Criterion 3.2.4, replacing “set of web pages” with “set of non-web documents or a set of software programs”.

With these substitutions, it would read:

3.2.4 Consistent Identification: Components that have the same functionality within a [**set of non-web documents** or a **set of software programs**] are identified consistently.

NOTE 1 (ADDED)

See set of documents and set of software programs in the Key Terms section to determine when a group of documents or software programs is considered a set for this success criterion. Those implementing this document (WCAG2ICT) will need to consider if this success criterion is appropriate to apply to non-web documents and software. See the Interpretation of Web Terminology in a Non-web Context.

NOTE 2 (ADDED)

Although not required by this success criterion, ensuring that component identification be consistent when they occur more than once *within* non-web documents or software programs directly addresses user needs identified in the Intent section for this success criterion, and is generally considered best practice.

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

Sets of software that meet this definition appear to be extremely rare.

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *3.2.6 Consistent Help*

(Level A)

If a web page contains any of the following help mechanisms, and those mechanisms are repeated on multiple web pages within a set of web pages, they occur in the same order relative to other page content, unless a change is initiated by the user:

- Human contact details;
- Human contact mechanism;
- Self-help option;
- A fully automated contact mechanism.

NOTE 1

Help mechanisms may be provided directly on the page, or may be provided via a direct link to a different page containing the information.

NOTE 2

For this success criterion, "the same order relative to other page content" can be thought of as how the content is ordered when the page is serialized. The visual position of a help mechanism is likely to be consistent across pages for the same page variation (e.g., CSS break-point). The user can initiate a change, such as changing the page's zoom or orientation, which may trigger a different page variation. This criterion is concerned with relative order across pages displayed in the same page variation (e.g., same zoom level and orientation).

§ Applying SC 3.2.6 Consistent Help to Non-Web Documents and Software

This applies directly as written and as described in Intent from Understanding Success Criterion 3.2.6, replacing "web page(s)" and "page(s)" with "non-web document(s) or software program(s)", "set of web pages" with "set of non-web documents or set of software programs", "page content" with "content", "on the page" with "in the non-web document or software", "page is serialized" with "non-

web document or software content is serialized", "different page" with "different non-web document, software, or web page", and "page variation" with "content layout variation".

With these substitutions, it would read:

3.2.6 Consistent Help: If a [**non-web document** or **software**] contains any of the following help mechanisms, and those mechanisms are repeated [in multiple non-web documents or software] within a [**set of non-web documents** or **set of software programs**], they occur in the same order relative to other [**content**], unless a change is initiated by the user:

- Human contact details;
- Human contact mechanism;
- Self-help option;
- A fully automated contact mechanism

> NOTE 1
>
> Help mechanisms may be provided directly [in the non-web document or software], or may be provided via a direct link to a [different non-web document, software, or web page] containing the information.

> NOTE 2
>
> For this success criterion, "the same order relative to other [content]" can be thought of as how the content is ordered when the [non-web document or software content is serialized]. The visual position of a help mechanism is likely to be consistent across [non-web documents or software] for the same [content layout variation] (e.g., CSS break-point). The user can initiate a change, such as changing the [non-web document's or software's] zoom or orientation, which may trigger a different [content layout variation]. This criterion is concerned with relative order across [non-web documents or software] displayed in the same [content layout variation] (e.g., same zoom level and orientation).

NOTE 3 (ADDED)

See set of documents and set of software programs in the Key Terms section to determine when a group of documents or software programs is considered a set for this success criterion. Those implementing this document (WCAG2ICT) will need to consider if this success criterion is appropriate to apply to non-web documents and software. See the Interpretation of Web Terminology in a Non-web Context.

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)
Sets of software that meet this definition appear to be extremely rare.

§ 3.3 Input Assistance

> Help users avoid and correct mistakes.

§ *Applying Guideline 3.3 Input Assistance to Non-Web Documents and Software*

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 3.3 applies directly as written.

§ *3.3.1 Error Identification*

(Level A)

> If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text.

§ Applying SC 3.3.1 Error Identification to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.3.1.

> NOTE (ADDED) (FOR NON-WEB SOFTWARE)
>
> See also the Comments on Closed Functionality.

§ *3.3.2 Labels or Instructions*

(Level A)

> Labels or instructions are provided when content requires user input.

§ Applying SC 3.3.2 Labels or Instructions to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.3.2.

§ *3.3.3 Error Suggestion*

(Level AA)

> If an input error is automatically detected and suggestions for correction are known, then the suggestions are provided to the user, unless it would jeopardize the security or purpose of the content.

§ Applying SC 3.3.3 Error Suggestion to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.3.3.

§ *3.3.4 Error Prevention (Legal, Financial, Data)*

(Level AA)

> For web pages that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true:
>
> **Reversible**
> Submissions are reversible.
>
> **Checked**
> Data entered by the user is checked for input errors and the user is provided an opportunity to correct them.
>
> **Confirmed**
> A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.

§ Applying SC 3.3.4 Error Prevention (Legal, Financial, Data) to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.3.4 replacing “web pages” with “non-web documents or software”.

With this substitution, it would read:

3.3.4 Error Prevention (Legal, Financial, Data): For **[non-web documents or software]** that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true:

Reversible
Submissions are reversible.

Checked
Data entered by the user is checked for input errors and the user is provided an opportunity to correct them.

Confirmed
A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.

§ *3.3.7 Redundant Entry*

(Level A)

> Information previously entered by or provided to the user that is required to be entered again in the same process is either:
>
> - auto-populated, or
> - available for the user to select.
>
> Except when:
>
> - re-entering the information is essential,
> - the information is required to ensure the security of the content, or
> - previously entered information is no longer valid.

§ Applying SC 3.3.7 Redundant Entry to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.3.7.

3.3.8 Accessible Authentication (Minimum)

(Level AA)

> A cognitive function test (such as remembering a password or solving a puzzle) is not required for any step in an authentication process unless that step provides at least one of the following:
>
> **Alternative**
> Another authentication method that does not rely on a cognitive function test.
>
> **Mechanism**
> A mechanism is available to assist the user in completing the cognitive function test.
>
> **Object Recognition**
> The cognitive function test is to recognize objects.
>
> **Personal Content**
> The cognitive function test is to identify non-text content the user provided to the website.
>
> *NOTE 1*
>
> *"Object recognition" and "Personal content" may be represented by images, video, or audio.*
>
> *NOTE 2*
> *Examples of mechanisms that satisfy this criterion include:*
>
> - *support for password entry by password managers to reduce memory need, and*
> - *copy and paste to reduce the cognitive burden of re-typing.*

Applying SC 3.3.8 Accessible Authentication (Minimum) to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 3.3.8, replacing “the website” with “a website, non-web document, or software”.

With this substitution, it would read:

3.3.8 Accessible Authentication (Minimum): A cognitive function test (such as remembering a password or solving a puzzle) is not required for any step in an authentication process unless that step provides at least one of the following:

Alternative
Another authentication method that does not rely on a cognitive function test.

Mechanism
A mechanism is available to assist the user in completing the cognitive function test.

Object Recognition
The cognitive function test is to recognize objects.

Personal Content
The cognitive function test is to identify non-text content the user provided to [a website, **non-web document**, or **software**].

NOTE 1

"Object recognition" and "Personal content" may be represented by images, video, or audio.

NOTE 2

Examples of mechanisms that satisfy this criterion include:

1. support for password entry by password managers to reduce memory need, and
2. copy and paste to reduce the cognitive burden of re-typing.

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

Any passwords used to unlock underlying platform software (running below the non-web software) are out of scope for this requirement since these are not under control of the non-web software's author.

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

There are cases where non-web software has an authentication process and no alternative or assistance mechanism is feasible, for example when entering a password when starting, powering on / turning on an ICT (device or otherwise). In such situations, it may not be possible for the non-web software to satisfy this success criterion.

NOTE 5 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ 4. Robust

> Content must be robust enough that it can be interpreted by a wide variety of user agents, including assistive technologies.

§ Applying Principle 4 Robust to Non-Web Documents and Software

In WCAG 2, the Principles are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Principle 4 applies directly as written replacing “user agents, including assistive technologies” with “assistive technologies and accessibility features of software”.

With this substitution, it would read:

Principle 4 Robust: Content must be robust enough that it can be interpreted by a wide variety of [**assistive technologies** and accessibility features of software].

4.1 Compatible

> Maximize compatibility with current and future user agents, including assistive technologies.

Applying Guideline 4.1 Compatible to Non-Web Documents and Software

In WCAG 2, the Guidelines are provided for framing and understanding the success criteria under them but are not used for conformance to WCAG. Guideline 4.1 applies directly as written, replacing “user agents, including assistive technologies” with “assistive technologies and accessibility features of software”.

With this substitution, it would read:

Guideline 4.1 Compatible: Maximize compatibility with current and future [**assistive technologies** and accessibility features of software].

4.1.1 Parsing (WCAG 2.1)

(Level A)

In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features.

NOTE 1

This success criterion should be considered as always satisfied for any content using HTML or XML.

NOTE 2

Since this criterion was written, the HTML Living Standard has adopted specific requirements governing how user agents must handle incomplete tags, incorrect element nesting, duplicate attributes, and non-unique IDs. [HTML]

Although the HTML standard treats some of these cases as non-conforming for authors, it is considered to "allow these features" for the purposes of this success criterion because the specification requires that user agents support handling these cases consistently. In practice, this criterion no longer provides any benefit to people with disabilities in itself.

Issues such as missing roles due to inappropriately nested elements or incorrect states or names due to a duplicate ID are covered by different success criteria and should be reported under those criteria rather than as issues with 4.1.1.

§ Applying SC 4.1.1 Parsing (WCAG 2.0 and 2.1) to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 4.1.1, replacing “In content implemented using markup languages” with “For non-web documents or software that use markup languages, in such a way that the markup is separately exposed and available to assistive technologies and accessibility features of software or to a user-selectable user agent” and replacing the WCAG notes with notes applicable to non-web documents and software.

With this substitution, it would read:

4.1.1 Parsing: [For **non-web documents** or **software** that use markup languages, in such a way that the markup is separately exposed and available to **assistive technologies** and accessibility features of

software or to a user-selectable **user agent**], elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features.

NOTE 1 (ADDED)

Markup is not always available to assistive technologies or to user selectable user agents such as browsers. Software sometimes uses markup languages internally for persistence of the software user interface, in ways where the markup is never available to assistive technology (either directly or through a document object model (DOM)), or to a user agent (such as a browser). In such cases, conformance to this provision would have no impact on accessibility as it can have for web content where it is exposed.

Accessibility issues introduced through poor markup would surface as errors in the programmatic information and would be reported using success criteria that rely on that information, such as 1.3.1 Info and Relationships and 4.1.2 Name, Role, Value.

NOTE 2 (ADDED)

This success criterion would be satisfied in cases where:

- Content is implemented using HTML or XML (as outlined in the WCAG 2.1 note on 4.1.1 and the WCAG 2.0 editorial errata in the thirteenth list item)
- Non-web documents or software are not authored using a markup language.
- Non-web documents or software are authored using a markup language, but accessibility information is exposed via platform accessibility APIs, not by making the markup itself available to assistive technologies.

Example (Added): Examples where 4.1.1 Parsing would be satisfied:

- An HTML page embedded inside a desktop application (as outlined in the WCAG 2.1 note on 4.1.1 and the WCAG 2.0 editorial errata in the thirteenth list item)
- A PDF document (not authored using a markup language)
- Android or iOS apps which use a markup language to specify UI layout (accessibility information is exposed via platform accessibility APIs, not the markup)

Examples of markup that might be separately exposed and available to assistive technologies and to user agents include:

- LaTeX documents
- Markdown documents

NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *4.1.1 Parsing (WCAG 2.2)*

(Obsolete and removed)

NOTE

This criterion was originally adopted to address problems that assistive technology had directly parsing HTML. Assistive technology no longer has any need to directly parse HTML. Consequently, these problems either no longer exist or are addressed by other criteria. This criterion no longer has utility and is removed.

§ Applying SC 4.1.1 Parsing (Obsolete and removed) (WCAG 2.2) to Non-Web Documents and Software

> NOTE (ADDED)
>
> WCAG 2.2 has made this success criterion obsolete and removed it as a requirement in the standard. Therefore, the interpretation of this success criterion for non-web documents and software has been removed.

§ *4.1.2 Name, Role, Value*

(Level A)

> For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies.
>
> > *NOTE*
> >
> > *This success criterion is primarily for web authors who develop or script their own user interface components. For example, standard HTML controls already meet this success criterion when used according to specification.*

§ Applying SC 4.1.2 Name, Role, Value to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 4.1.2, replacing “user agents, including assistive technologies", with “assistive technologies and accessibility features of underlying software” and the note with: “This success criterion is primarily for software

developers who develop or use custom user interface components. For example, standard user interface components on most accessibility-supported platforms already satisfy this success criterion when used according to specification."

With this substitution, it would read:

4.1.2 Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to [**assistive technologies** and accessibility features of underlying **software**].

> NOTE 1 (ADDED) (FOR NON-WEB DOCUMENTS)
>
> For non-web document formats that support interoperability with assistive technology, standard user interface components often satisfy this success criterion when used according to the general design and accessibility guidance for the document format.

> NOTE 2 (REPLACED) (FOR NON-WEB SOFTWARE)
>
> This success criterion is primarily for software developers who develop or use custom user interface components. Standard user interface components on most accessibility-supported platforms already satisfy this success criterion when used according to specification.

> NOTE 3 (ADDED) (FOR NON-WEB SOFTWARE)
>
> For conforming to this success criterion, it is usually best practice for software user interfaces to use the accessibility services of platform software. These accessibility services enable interoperability between software user interfaces and both assistive technologies and accessibility features of software in standardized ways. Most platform accessibility services go beyond programmatic exposure of name and role, and programmatic setting of states, properties and values (and notification of same), and specify additional information that could be exposed and / or set (for instance, a list of the available actions for a given user interface component, and a means to programmatically execute one of the listed actions).

NOTE 4 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ *4.1.3 Status Messages*

(Level AA)

> In content implemented using markup languages, status messages can be programmatically determined through role or properties such that they can be presented to the user by assistive technologies without receiving focus.

§ Applying SC 4.1.3 Status Messages to Non-Web Documents and Software

This applies directly as written, and as described in Intent from Understanding Success Criterion 4.1.3.

NOTE 1 (ADDED)

For non-web documents and software where status messages are not implemented using markup languages, there is still a user need to have status messages be programmatically exposed so that they can be presented to the user by assistive technologies without receiving focus. This is typically enabled through the use of accessibility services of the user agent or other platform software.

NOTE 2 (ADDED) (FOR NON-WEB SOFTWARE)

See also the Comments on Closed Functionality.

§ Comments on Definitions in WCAG 2 Glossary

The sections that follow are organized according to the glossary from WCAG 2. The text of each term from WCAG 2 is copied as quoted text. Following that, the WCAG2ICT guidance is provided. The WCAG2ICT guidance can be found in the sections where the headings begin with "Applying..." to highlight that this is the content specific to this document. Within these sections custom notes added by WCAG2ICT are marked with the text "ADDED".

The following is a complete list of definitions from the WCAG 2 glossary. Some items apply to all technologies and do not require additional guidance in this document; guidance on the remainder follows.

§ Glossary Items that Apply to All Technologies

The following glossary items apply to all technologies and do not need further interpretation for non-web ICT.

- abbreviation
- alternative to time-based media
- ASCII art
- audio
- audio description
- audio-only
- blinking
- CAPTCHA
- captions
- conformance
- correct reading sequence
- dragging movements
- emergency

- essential
- extended audio description
- flash
- focus indicator
- functionality
- human language
- idiom
- image of text
- informative
- jargon
- legal commitments
- link purpose
- live
- lower secondary education level
- mechanism
- media alternative for text
- minimum bounding box
- navigated sequentially
- non-text content
- normative
- on a full-screen window
- paused
- pointer input
- prerecorded
- presentation
- primary education level
- process
- programmatically determined link context
- pure decoration

- real-time event
- relationships
- relied upon (technologies that are)
- same relative order
- sign language
- sign language interpretation
- single pointer
- specific sensory experience
- state
- status message
- synchronized media
- text
- text alternative
- used in an unusual or restricted way
- user-controllable
- video
- video-only
- visually customized

§ Glossary Items Used only in AAA Success Criteria

This document does not provide guidance on applying AAA success criteria to non-web ICT, including the following definitions.

- blocks of text
- context-sensitive help
- motion animation
- perimeter
- region
- section

- supplemental content
- user inactivity

§ Glossary Items with Specific Guidance

Additional guidance is provided for the following glossary entries from WCAG 2 when applying them to non-web documents and software.

§ **accessibility supported**

supported by users' assistive technologies as well as the accessibility features in browsers and other user agents

To qualify as an accessibility-supported use of a web content technology (or feature of a technology), both 1 and 2 must be satisfied for a web content technology (or feature):

1. **The way that the web content technology is used must be supported by users' assistive technology (AT).** This means that the way that the technology is used has been tested for interoperability with users' assistive technology in the human language(s) of the content,

   AND

2. **The web content technology must have accessibility-supported user agents that are available to users.** This means that at least one of the following four statements is true:

   1. The technology is supported natively in widely-distributed user agents that are also accessibility supported (such as HTML and CSS);

      OR

   2. The technology is supported in a widely-distributed plug-in that is also accessibility supported;

      OR

   3. The content is available in a closed environment, such as a university or corporate network, where the user agent required by the technology and used by the organization is also accessibility supported;

      OR

   4. The user agent(s) that support the technology are accessibility supported and are available for download or purchase in a way that:

      - does not cost a person with a disability any more than a person without a disability **and**
      - is as easy to find and obtain for a person with a disability as it is for a person without disabilities.

NOTE 1

The Accessibility Guidelines Working Group and the W3C do not specify which or how much support by assistive technologies there must be for a particular use of a web technology in order for it to be classified as accessibility supported. (See Level of Assistive Technology Support Needed for "Accessibility Support".)

NOTE 2

Web technologies can be used in ways that are not accessibility supported as long as they are not relied upon and the page as a whole meets the conformance requirements, including Conformance Requirement 4 and Conformance Requirement 5.

NOTE 3

When a web technology is used in a way that is "accessibility supported," it does not imply that the entire technology or all uses of the technology are supported. Most technologies, including HTML, lack support for at least one feature or use. Pages conform to WCAG only if the uses of the technology that are accessibility supported can be relied upon to meet WCAG requirements.

NOTE 4

When citing web content technologies that have multiple versions, the version(s) supported should be specified.

NOTE 5

One way for authors to locate uses of a technology that are accessibility supported would be to consult compilations of uses that are documented to be accessibility supported. (See Understanding Accessibility-Supported Web Technology Uses.) Authors, companies, technology vendors, or others may document accessibility-supported ways of using web content technologies. However, all ways of using technologies in the documentation would need to meet the definition of accessibility-supported Web content technologies above.

§ *Applying "accessibility supported" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "browsers and other user agents" with "user agents or other software", replacing "user agents" with "user agents or other software", replacing "web content technology" with "non-web document or software technology", adding "or other software extension" after "plug-in", and replacing all five of the Notes with a single Note: "Note: The concepts behind the five Notes and in Understanding Accessibility Supported are applicable to web technologies. The same or similar factors are applicable for non-web technologies."

With these substitutions and addition, it would read:

accessibility supported

supported by users' assistive technologies as well as the accessibility features in [**user agents** or other **software**]

To qualify as an accessibility-supported use of a [**non-web document** or software] technology (or feature of a technology), both 1 and 2 must be satisfied for a [non-web document or software] technology (or feature):

1. **The way that the [non-web document or software technology] is used must be supported by users' assistive technology (AT).** This means that the way that the technology is used has been tested for interoperability with users' assistive technology in the human language(s) of the content,

   AND

2. **The [non-web document or software] technology must have accessibility-supported user agents [or other software] that are available to users.** This means that at least one of the following four statements is true:

   1. The technology is supported natively in widely-distributed user agents [or other software] that are also accessibility supported (such as HTML and CSS);

      OR

   2. The technology is supported in a widely-distributed plug-in [or other software extension] that is also accessibility supported;

      OR

3. The content is available in a closed environment, such as a university or corporate network, where the user agent [or other software] required by the technology and used by the organization is also accessibility supported;

   OR

4. The user agent(s) that support the technology are accessibility supported and are available for download or purchase in a way that:

   - does not cost a person with a disability any more than a person without a disability **and**
   - is as easy to find and obtain for a person with a disability as it is for a person without disabilities.

> NOTE (REPLACED)
>
> The concepts behind the five Notes and in Understanding Accessibility Supported are applicable to web technologies. The same or similar factors are applicable for non-web technologies.

§ **ambiguous to users in general**

> the purpose cannot be determined from the link and all information of the web page presented to the user simultaneously with the link (i.e., readers without disabilities would not know what a link would do until they activated it)
>
> *EXAMPLE*
>
> *Example: The word guava in the following sentence "One of the notable exports is guava" is a link. The link could lead to a definition of guava, a chart listing the quantity of guava exported or a photograph of people harvesting guava. Until the link is activated, all readers are unsure and the person with a disability is not at any disadvantage.*

§ *Applying "ambiguous to users in general" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "web page" with "non-web document or software".

With this substitution, it would read:

ambiguous to users in general
the purpose cannot be determined from the link and all information of the **[non-web document or software]** presented to the user simultaneously with the link (i.e., readers without disabilities would not know what a link would do until they activated it)

> Example: The word guava in the following sentence "One of the notable exports is guava" is a link. The link could lead to a definition of guava, a chart listing the quantity of guava exported or a photograph of people harvesting guava. Until the link is activated, all readers are unsure and the person with a disability is not at any disadvantage.

§ **assistive technology**

hardware and/or software that acts as a user agent, or along with a mainstream user agent, to provide functionality to meet the requirements of users with disabilities that go beyond those offered by mainstream user agents

NOTE 1

Functionality provided by assistive technology includes alternative presentations (e.g., as synthesized speech or magnified content), alternative input methods (e.g., voice), additional navigation or orientation mechanisms, and content transformations (e.g., to make tables more accessible).

NOTE 2

Assistive technologies often communicate data and messages with mainstream user agents by using and monitoring APIs.

NOTE 3

The distinction between mainstream user agents and assistive technologies is not absolute. Many mainstream user agents provide some features to assist individuals with disabilities. The basic difference is that mainstream user agents target broad and diverse audiences that usually include people with and without disabilities. Assistive technologies target narrowly defined populations of users with specific disabilities. The assistance provided by an assistive technology is more specific and appropriate to the needs of its target users. The mainstream user agent may provide important functionality to assistive technologies like retrieving web content from program objects or parsing markup into identifiable bundles.

EXAMPLE

Example: Assistive technologies that are important in the context of this document include the following:

- *screen magnifiers, and other visual reading assistants, which are used by people with visual, perceptual and physical print disabilities to change text font, size, spacing, color, synchronization with speech, etc. in order to improve the visual readability of rendered text and images;*
- *screen readers, which are used by people who are blind to read textual information through synthesized speech or braille;*
- *text-to-speech software, which is used by some people with cognitive, language, and learning disabilities to convert text into synthetic speech;*
- *speech recognition software, which may be used by people who have some physical disabilities;*
- *alternative keyboards, which are used by people with certain physical disabilities to simulate the keyboard (including alternate keyboards that use head pointers, single switches, sip/puff and other special input devices.);*
- *alternative pointing devices, which are used by people with certain physical disabilities to simulate mouse pointing and button activations.*

§ *Applying "assistive technology" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "acts as a user agent" with "acts stand-alone", replacing "a mainstream user agent" with "mainstream information and communication technologies (ICT)" (later "mainstream ICT"), and replacing "web content" with "content".

With these substitutions, it would read:

assistive technology (as used in this document)
: hardware and/or software that acts [stand-alone], or along with [mainstream information and communication technologies (ICT)], to provide functionality to meet the requirements of users with disabilities that go beyond those offered by [mainstream ICT]

NOTE 1

Functionality provided by assistive technology includes alternative presentations (e.g., as synthesized speech or magnified content), alternative input methods (e.g., voice), additional navigation or orientation mechanisms, and content transformations (e.g., to make tables more accessible).

NOTE 2

Assistive technologies often communicate data and messages with [mainstream ICTs] by using and monitoring APIs.

NOTE 3

The distinction between [mainstream ICTs] and assistive technologies is not absolute. Many [mainstream ICTs] provide some features to assist individuals with disabilities. The basic difference is that [mainstream ICTs] target broad and diverse audiences that usually include people with and without disabilities. Assistive technologies target narrowly defined populations of users with specific disabilities. The assistance provided by an assistive technology is more specific and appropriate to the needs of its target users. The [mainstream ICT] may provide important functionality to assistive technologies like retrieving [**content**] from program objects or parsing markup into identifiable bundles.

Example: Assistive technologies that are important in the context of this document include the following:

- screen magnifiers, and other visual reading assistants, which are used by people with visual, perceptual and physical print disabilities to change text font, size, spacing, color, synchronization with speech, etc. in order to improve the visual readability of rendered text and images;
- screen readers, which are used by people who are blind to read textual information through synthesized speech or braille;
- text-to-speech software, which is used by some people with cognitive, language, and learning disabilities to convert text into synthetic speech;
- speech recognition software, which may be used by people who have some physical disabilities;
- alternative keyboards, which are used by people with certain physical disabilities to simulate the keyboard (including alternate keyboards that use head pointers, single switches, sip/puff and other special input devices.);
- alternative pointing devices, which are used by people with certain physical disabilities to simulate mouse pointing and button activations.

§ **changes of context**

major changes that, if made without user awareness, can disorient users who are not able to view the entire page simultaneously

Changes in context include changes of:

- user agent;
- viewport;
- focus;
- content that changes the meaning of the web page

NOTE

A change of content is not always a change of context. Changes in content, such as an expanding outline, dynamic menu, or a tab control do not necessarily change the context, unless they also change one of the above (e.g., focus).

EXAMPLE

Example: Opening a new window, moving focus to a different component, going to a new page (including anything that would look to a user as if they had moved to a new page) or significantly re-arranging the content of a page are examples of changes of context.

§ *Applying "changes of context" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "page" with "non-web document or content presented by software".

With these substitutions, it would read:

changes of context

major changes that, if made without user awareness, can disorient users who are not able to view the entire [**non-web document** or **content** presented by **software**] simultaneously

Changes in context include changes of:

1. user agent;
2. viewport;
3. focus;
4. content that changes the meaning of the [non-web document or content presented by software].

> NOTE 1
>
> A change of content is not always a change of context. Changes in content, such as an expanding outline, dynamic menu, or a tab control do not necessarily change the context, unless they also change one of the above (e.g., focus).

> Example: Opening a new window, moving focus to a different component, going to a new page (including anything that would look to a user as if they had moved to a new page) or significantly re-arranging the content of a page are examples of changes of context.

> NOTE 2 (ADDED)
>
> A change in the user agent might include bringing up a new window, or might be a significant change in the menus and/or toolbars that are displayed and available for interacting with some portion of the document.

§ **cognitive function test**

> A task that requires the user to remember, manipulate, or transcribe information. Examples include, but are not limited to:
>
> - memorization, such as remembering a username, password, set of characters, images, or patterns. The common identifiers name, e-mail, and phone number are not considered cognitive function tests as they are personal to the user and consistent across websites;
> - transcription, such as typing in characters;
> - use of correct spelling;
> - performance of calculations;
> - solving of puzzles.

§ *Applying "cognitive function test" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "websites" with "websites, non-web documents, and software".

With this substitution, it would read:

cognitive function test

A task that requires the user to remember, manipulate, or transcribe information. Examples include, but are not limited to:

memorization, such as remembering a username, password, set of characters, images, or patterns. The common identifiers name, e-mail, and phone number are not considered cognitive function tests as they are personal to the user and consistent across [websites, **non-web documents**, and **software**];

- transcription, such as typing in characters;
- use of correct spelling;
- performance of calculations;
- solving of puzzles.

§ **conforming alternate version**

version that

1. conforms at the designated level, and
2. provides all of the same information and functionality in the same human language, and
3. is as up to date as the non-conforming content, and
4. for which at least one of the following is true:
    1. the conforming version can be reached from the non-conforming page via an accessibility-supported mechanism, or
    2. the non-conforming version can only be reached from the conforming version, or
    3. the non-conforming version can only be reached from a conforming page that also provides a mechanism to reach the conforming version

NOTE 1

In this definition, "can only be reached" means that there is some mechanism, such as a conditional redirect, that prevents a user from "reaching" (loading) the non-conforming page unless the user had just come from the conforming version.

NOTE 2

The alternate version does not need to be matched page for page with the original (e.g., the conforming alternate version may consist of multiple pages).

NOTE 3

If multiple language versions are available, then conforming alternate versions are required for each language offered.

NOTE 4

Alternate versions may be provided to accommodate different technology environments or user groups. Each version should be as conformant as possible. One version would need to be fully conformant in order to meet conformance requirement 1.

NOTE 5

The conforming alternative version does not need to reside within the scope of conformance, or even on the same website, as long as it is as freely available as the non-conforming version.

NOTE 6

Alternate versions should not be confused with supplementary content, which support the original page and enhance comprehension.

NOTE 7

Setting user preferences within the content to produce a conforming version is an acceptable mechanism for reaching another version as long as the method used to set the preferences is accessibility supported.

See Understanding Conforming Alternate Versions

§ *Applying "conforming alternate version" to Non-Web Documents and Software*

The guidance in this document does not use the term "conforming alternate version".

See the section Comments on Conformance.

§ **content**

> information and sensory experience to be communicated to the user by means of a user agent, including code or markup that defines the content's structure, presentation, and interactions

§ *Applying "content (Web Content)" to Non-Web Documents and Software*

See the guidance on content in the Key Terms section.

§ **contrast ratio**

(L1 + 0.05) / (L2 + 0.05), where

- L1 is the relative luminance of the lighter of the colors, and
- L2 is the relative luminance of the darker of the colors.

NOTE 1

Contrast ratios can range from 1 to 21 (commonly written 1:1 to 21:1).

NOTE 2

Because authors do not have control over user settings as to how text is rendered (for example font smoothing or anti-aliasing), the contrast ratio for text can be evaluated with anti-aliasing turned off.

NOTE 3

For the purpose of Success Criteria 1.4.3 and 1.4.6, contrast is measured with respect to the specified background over which the text is rendered in normal usage. If no background color is specified, then white is assumed.

NOTE 4

Background color is the specified color of content over which the text is to be rendered in normal usage. It is a failure if no background color is specified when the text color is specified, because the user's default background color is unknown and cannot be evaluated for sufficient contrast. For the same reason, it is a failure if no text color is specified when a background color is specified.

NOTE 5

When there is a border around the letter, the border can add contrast and would be used in calculating the contrast between the letter and its background. A narrow border around the letter would be used as the letter. A wide border around the letter that fills in the inner details of the letters acts as a halo and would be considered background.

NOTE 6

WCAG conformance should be evaluated for color pairs specified in the content that an author would expect to appear adjacent in typical presentation. Authors need not consider unusual presentations, such as color changes made by the user agent, except where caused by authors' code.

§ *Applying "contrast ratio" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary.

NOTE (ADDED)

Because relative luminance is defined such that it cannot directly apply to hardware, please note the text in the introduction which reads: "This document does not comment on hardware aspects of products, because the basic constructs on which WCAG 2 is built do not apply to these."

§ CSS pixel

> visual angle of about 0.0213 degrees
>
> A CSS pixel is the canonical unit of measure for all lengths and measurements in CSS. This unit is density-independent, and distinct from actual hardware pixels present in a display. User agents and operating systems should ensure that a CSS pixel is set as closely as possible to the CSS Values and Units Module Level 3 reference pixel [*css3-values*], which takes into account the physical dimensions of the display and the assumed viewing distance (factors that cannot be determined by content authors).

§ *Applying "CSS pixel" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary.

> NOTE 1 (ADDED)
>
> Non-web software and its accompanying platform software do not use CSS pixel measurements. Therefore, use platform-defined density-independent pixel measurements which approximate the CSS reference pixel. Examples of platform-defined density-independent pixel measurements include: points (pt) for iOS and macOS, density-independent pixels (dp) for Android, and effective pixels (epx) for Windows.

NOTE 2 (ADDED)

Examples where a density-independent pixel may not be defined in the platform:

- Software designed for specific hardware, such as kiosks or office equipment, where the author knows the physical screen size and, potentially, the pixel density.
- Software, such as streaming apps on smart TV platforms or similar software, where the author may lack information about the physical screen size but may know an appropriate viewing distance or viewing angle.

When there is no platform-defined density-independent pixel measurement, the reference pixel size can be approximated in the following manner:

- Determine a viewing distance that matches the use case and display type. For instance, in the case of a touchscreen, the viewing distance is normally less than the length of an arm, typically around 28 inches (71 cm).
- Calculate the size of the reference pixel: Divide the viewing distance by 2688. The number 2688 is obtained by dividing 28 inches (arm's length) by the derived reference pixel size (1/96 inch).

NOTE 3 (ADDED)

Most software and devices are usable at more than one viewing distance. However, only viewing distances that are plausible for the product can be considered an appropriate approximation for the reference pixel. For example, in software designed for use with a touchscreen, a visual-angle pixel longer than 0.11 inch (0.28 mm) would not be plausible, because this would signify a viewing distance of more than arm’s length.

NOTE 4 (ADDED)

People with low vision often use devices at less than the standard viewing distance. However, basing the density-independent pixel on a typical viewing distance provides a balance of benefits for users with disabilities. If a longer viewing distance were chosen as the basis for the density-independent pixel, the viewport would be measured with a smaller number of larger pixels, causing success criterion 1.4.10 Reflow to be less stringent. If a shorter viewing distance were chosen, user interface components would be measured with a larger number of smaller pixels, causing some success criteria, such as 2.5.8 Target Size, to be less stringent.

§ **down-event**

> platform event that occurs when the trigger stimulus of a pointer is depressed
>
> The down-event may have different names on different platforms, such as "touchstart" or "mousedown".

§ *Applying “down-event” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary.

> NOTE (ADDED)
>
> The down-event may have different names on different platforms. For example ["PointerPressed” or “mousedown”].

§ **general flash and red flash thresholds**

a flash or rapidly changing image sequence is below the threshold (i.e., content **passes**) if any of the following are true:

- there are no more than three **general flashes** and / or no more than three **red flashes** within any one-second period; or
- the combined area of flashes occurring concurrently occupies no more than a total of .006 steradians within any 10 degree visual field on the screen (25% of any 10 degree visual field on the screen) at typical viewing distance

where:

- A **general flash** is defined as a pair of opposing changes in relative luminance of 10% or more of the maximum relative luminance (1.0) where the relative luminance of the darker image is below 0.80; and where "a pair of opposing changes" is an increase followed by a decrease, or a decrease followed by an increase, and
- A **red flash** is defined as any pair of opposing transitions involving a saturated red

Exception: Flashing that is a fine, balanced, pattern such as white noise or an alternating checkerboard pattern with "squares" smaller than 0.1 degree (of visual field at typical viewing distance) on a side does not violate the thresholds.

NOTE 1

For general software or web content, using a 341 x 256 pixel rectangle anywhere on the displayed screen area when the content is viewed at 1024 x 768 pixels will provide a good estimate of a 10 degree visual field for standard screen sizes and viewing distances (e.g., 15-17 inch screen at 22-26 inches). This resolution of 75 - 85 ppi is known to be lower, and thus more conservative than the nominal CSS pixel resolution of 96 ppi in CSS specifications. Higher resolutions displays showing the same rendering of the content yield smaller and safer images so it is lower resolutions that are used to define the thresholds.

NOTE 2

A transition is the change in relative luminance (or relative luminance/color for red flashing) between adjacent peaks and valleys in a plot of relative luminance (or relative luminance/color for red flashing) measurement against time. A flash consists of two opposing transitions.

NOTE 3

*The new working definition in the field for "**pair of opposing transitions involving a saturated red**" (from WCAG 2.2) is a pair of opposing transitions where, one transition is either to or from a state with a value R/(R + G + B) that is greater than or equal to 0.8, and the difference between states is more than 0.2 (unitless) in the CIE 1976 UCS chromaticity diagram. [ISO_9241-391]*

NOTE 4

Tools are available that will carry out analysis from video screen capture. However, no tool is necessary to evaluate for this condition if flashing is less than or equal to 3 flashes in any one second. Content automatically passes (see #1 and #2 above).

§ *Applying "general flash and red flash thresholds" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary.

NOTE (ADDED)

Because this deals with relative luminance and not luminance, it can only be applied to information on a display, not to hardware sources of light.

§ **input error**

> information provided by the user that is not accepted
>
> *NOTE*
>
> *This includes:*
>
> 1. *Information that is required by the web page but omitted by the user*
> 2. *Information that is provided by the user but that falls outside the required data format or values*

§ *Applying "input error" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "web page" with "non-web document or software".

With this substitution, it would read:

input error
information provided by the user that is not accepted

> NOTE
> This includes:
>
> 1. Information that is required by the [**non-web document** or **software**] but omitted by the user
> 2. Information that is provided by the user but that falls outside the required data format or values

§ **keyboard interface**

interface used by software to obtain keystroke input

NOTE 1

A keyboard interface allows users to provide keystroke input to programs even if the native technology does not contain a keyboard.

EXAMPLE

Example: A touchscreen PDA has a keyboard interface built into its operating system as well as a connector for external keyboards. Applications on the PDA can use the interface to obtain keyboard input either from an external keyboard or from other applications that provide simulated keyboard output, such as handwriting interpreters or speech-to-text applications with "keyboard emulation" functionality.

NOTE 2

Operation of the application (or parts of the application) through a keyboard-operated mouse emulator, such as MouseKeys, does not qualify as operation through a keyboard interface because operation of the program is through its pointing device interface, not through its keyboard interface.

§ *Applying “keyboard interface” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary.

NOTE 1 (ADDED)

Keyboard interface does not refer to a physical device but to the interface between the software and any keyboard or keyboard substitute (i.e., the interface where the software accepts text or other keystroke input from the platform which may come from a keyboard or from a keyboard alternative). Platform software may provide device independent input services to applications that enable operation via such a keyboard interface.

NOTE 2 (ADDED)

This success criterion does not imply that software always needs to directly support a keyboard or "keyboard interface". Nor does it imply that software always needs to provide a virtual keyboard.

§ **keyboard shortcut**

> alternative means of triggering an action by the pressing of one or more keys

§ *Applying "keyboard shortcut" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary.

NOTE (ADDED)

A key command issued by a long press of a key (2 seconds or more) and other accessibility features provided by the platform are not considered a keyboard shortcut. Such commands often occur when there are limited keys, or no modifier keys, present on a device.

§ **label**

> text or other component with a text alternative that is presented to a user to identify a component within web content
>
> *NOTE 1*
>
> *A label is presented to all users whereas the name may be hidden and only exposed by assistive technology. In many (but not all) cases the name and the label are the same.*
>
> *NOTE 2*
>
> *The term label is not limited to the label element in HTML.*

§ *Applying "label" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "Web Content" with "content" and adding "or by accessibility features of software" after "assistive technology" in Note 1.

With this substitution and addition, it would read:

label
text or other component with a text alternative that is presented to a user to identify a component within [**content**]

> NOTE 1
>
> A label is presented to all users whereas the name may be hidden and only exposed by assistive technology [or by accessibility features of **software**]. In many (but not all) cases the name and the label are the same.

NOTE 2

The term label is not limited to the label element in HTML.

§ **large scale**

with at least 18 point or 14 point bold or font size that would yield equivalent size for Chinese, Japanese and Korean (CJK) fonts

NOTE 1

Fonts with extraordinarily thin strokes or unusual features and characteristics that reduce the familiarity of their letter forms are harder to read, especially at lower contrast levels.

NOTE 2

Font size is the size when the content is delivered. It does not include resizing that may be done by a user.

NOTE 3

The actual size of the character that a user sees is dependent both on the author-defined size and the user's display or user agent settings. For many mainstream body text fonts, 14 and 18 point is roughly equivalent to 1.2 and 1.5 em or to 120% or 150% of the default size for body text (assuming that the body font is 100%), but authors would need to check this for the particular fonts in use. When fonts are defined in relative units, the actual point size is calculated by the user agent for display. The point size should be obtained from the user agent, or calculated based on font metrics as the user agent does, when evaluating this success criterion. Users who have low vision would be responsible for choosing appropriate settings.

NOTE 4

When using text without specifying the font size, the smallest font size used on major browsers for unspecified text would be a reasonable size to assume for the font. If a level 1 heading is rendered in 14pt bold or higher on major browsers, then it would be reasonable to assume it is large text. Relative scaling can be calculated from the default sizes in a similar fashion.

NOTE 5

The 18 and 14 point sizes for roman texts are taken from the minimum size for large print (14pt) and the larger standard font size (18pt). For other fonts such as CJK languages, the "equivalent" sizes would be the minimum large print size used for those languages and the next larger standard large print size.

§ *Applying “large scale (text)” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing “user agent” with “user agent or non-web software” in Note 3 and “browsers” with "browsers, user agents, or other platform software” in Note 4.

With this substitution and addition, it would read:

large scale (text)
with at least 18 point or 14 point bold or font size that would yield equivalent size for Chinese, Japanese and Korean (CJK) fonts

NOTE 1

Fonts with extraordinarily thin strokes or unusual features and characteristics that reduce the familiarity of their letter forms are harder to read, especially at lower contrast levels.

Font size is the size when the content is delivered. It does not include resizing that may be done by a user.

NOTE 2

The actual size of the character that a user sees is dependent both on the author-defined size and the user's display, [user agent or **non-web software**] settings. For many mainstream body text fonts, 14 and 18 point is roughly equivalent to 1.2 and 1.5 em or to 120% or 150% of the default size for body text (assuming that the body font is 100%), but authors would need to check this for the particular fonts in use. When fonts are defined in relative units, the actual point size is calculated by the [user agent or non-web software] for display. The point size should be obtained from the [user agent or non-web software], or calculated based on font metrics as the [user agent or non-web software] does, when evaluating this success criterion. Users who have low vision would be responsible for choosing appropriate settings.

NOTE 3

When using text without specifying the font size, the smallest font size used on major [browsers, **user agents**, or other **platform software**] for unspecified text would be a reasonable size to assume for the font. If a level 1 heading is rendered in 14pt bold or higher on major [browsers, user agents, or other platform software], then it would be reasonable to assume it is large text. Relative scaling can be calculated from the default sizes in a similar fashion.

NOTE 4 (ADDED)

When evaluating non-web documents and software, 1 point means 1.333 CSS pixels.

§ **name**

> text by which software can identify a component within web content to the user
>
> > *NOTE 1*
> >
> > *The name may be hidden and only exposed by assistive technology, whereas a label is presented to all users. In many (but not all) cases, the label and the name are the same.*
>
> > *NOTE 2*
> >
> > *This is unrelated to the name attribute in HTML.*

§ *Applying "name" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "web content" with "content" and adding "or by accessibility features of software" after "assistive technology" in Note 1.

With this substitution and addition, it would read:

name
text by which software can identify a component within [**content**] to the user

> NOTE 1
>
> The name may be hidden and only exposed by assistive technology [or by accessibility features of **software**], whereas a label is presented to all users. In many (but not all) cases, the label and the name are the same.

> NOTE 2
>
> This is unrelated to the name attribute in HTML.

> Example: For non-web software, “AccessibleName” (or the corresponding term used in different APIs) of the Accessibility API of the platform is an example of such a name.

§ **programmatically determined**

> determined by software from author-supplied data provided in a way that different user agents, including assistive technologies, can extract and present this information to users in different modalities
>
> > *EXAMPLE 1*
> >
> > *Example 1: Determined in a markup language from elements and attributes that are accessed directly by commonly available assistive technology.*
>
> > *EXAMPLE 2*
> >
> > *Example 2: Determined from technology-specific data structures in a non-markup language and exposed to assistive technology via an accessibility API that is supported by commonly available assistive technology.*

§ *Applying “programmatically determined” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing “user agents, including assistive technologies", with “assistive technologies and accessibility features of software” and adding “and accessibility features of software” after “assistive technology”.

With this substitution and addition, it would read:

programmatically determined (programmatically determinable)
: determined by software from author-supplied data provided in a way that different [**assistive technologies** and accessibility features of **software**] can extract and present this information to

users in different modalities

> Example 1: Determined in a markup language from elements and attributes that are accessed directly by commonly available assistive technology [and accessibility features of software].

> Example 2: Determined from technology-specific data structures in a non-markup language and exposed to assistive technology [and accessibility features of software] via an accessibility API that is supported by commonly available assistive technology [and accessibility features of software].

> NOTE (ADDED)
>
> Software typically enables content to be programmatically determined through the use of accessibility services of platform software. Non-web documents typically enable content to be programmatically determined through the use of accessibility services of the user agent.

§ **programmatically set**

> set by software using methods that are supported by user agents, including assistive technologies

§ *Applying "programmatically set" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "user agents, including assistive technologies" with "assistive technologies and accessibility features of software".

With this substitution, it would read:

programmatically set
set by software using methods that are supported by [**assistive technologies** and accessibility features of **software**]

NOTE (ADDED)

Software typically enables content to be programmatically determined through the use of accessibility services of platform software. Non-web documents typically enable content to be programmatically determined through the use of accessibility services of the user agent.

§ **relative luminance**

the relative brightness of any point in a colorspace, normalized to 0 for darkest black and 1 for lightest white

NOTE 1

*For the sRGB colorspace, the relative luminance of a color is defined as L = 0.2126 * **R** + 0.7152 * **G** + 0.0722 * **B** where **R**, **G** and **B** are defined as:*

- *if RsRGB <= 0.04045 then **R** = RsRGB/12.92 else **R** = ((RsRGB+0.055)/1.055) ^ 2.4*
- *if GsRGB <= 0.04045 then **G** = GsRGB/12.92 else **G** = ((GsRGB+0.055)/1.055) ^ 2.4*
- *if BsRGB <= 0.04045 then **B** = BsRGB/12.92 else **B** = ((BsRGB+0.055)/1.055) ^ 2.4*

and RsRGB, GsRGB, and BsRGB are defined as:

- *RsRGB = R8bit/255*
- *GsRGB = G8bit/255*
- *BsRGB = B8bit/255*

The "^" character is the exponentiation operator. (Formula taken from [SRGB].)

NOTE 2

Before May 2021 the value of 0.04045 in the definition was different (0.03928). It was taken from an older version of the specification and has been updated. It has no practical effect on the calculations in the context of these guidelines.

NOTE 3

Almost all systems used today to view web content assume sRGB encoding. Unless it is known that another color space will be used to process and display the content, authors should evaluate using sRGB colorspace. If using other color spaces, see Understanding Success Criterion 1.4.3.

NOTE 4

If dithering occurs after delivery, then the source color value is used. For colors that are dithered at the source, the average values of the colors that are dithered should be used (average R, average G, and average B).

NOTE 5

Tools are available that automatically do the calculations when testing contrast and flash.

NOTE 6

A separate page giving the relative luminance definition using MathML to display the formulas is available.

§ *Applying "relative luminance" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "web content" with "content".

With this substitution, it would read:

relative luminance
: the relative brightness of any point in a colorspace, normalized to 0 for darkest black and 1 for lightest white

NOTE 1

For the sRGB colorspace, the relative luminance of a color is defined as L = 0.2126 * **R** + 0.7152 * **G** + 0.0722 * **B** where **R**, **G** and **B** are defined as:

- if RsRGB <= 0.04045 then **R** = RsRGB/12.92 else **R** = ((RsRGB+0.055)/1.055) ^ 2.4
- if GsRGB <= 0.04045 then **G** = GsRGB/12.92 else **G** = ((GsRGB+0.055)/1.055) ^ 2.4
- if BsRGB <= 0.04045 then **B** = BsRGB/12.92 else **B** = ((BsRGB+0.055)/1.055) ^ 2.4

and RsRGB, GsRGB, and BsRGB are defined as:

- RsRGB = R8bit/255
- GsRGB = G8bit/255
- BsRGB = B8bit/255

The "^" character is the exponentiation operator. (Formula taken from [sRGB].)

NOTE 2

Before May 2021 the value of 0.04045 in the definition was different (0.03928). It was taken from an older version of the specification and has been updated. It has no practical effect on the calculations in the context of these guidelines.

NOTE 3

Almost all systems used today to view **[content]** assume sRGB encoding. Unless it is known that another color space will be used to process and display the content, authors should evaluate using sRGB colorspace. If using other color spaces, see Understanding Success Criterion 1.4.3.

NOTE 4

If dithering occurs after delivery, then the source color value is used. For colors that are dithered at the source, the average values of the colors that are dithered should be used (average R, average G, and average B).

NOTE 5

Tools are available that automatically do the calculations when testing contrast and flash.

NOTE 6

A separate page giving the relative luminance definition using MathML to display the formulas is available.

NOTE 7 (ADDED)

Because relative luminance is defined such that it cannot directly apply to hardware, please note the text in the introduction which reads: “This document does not comment on hardware aspects of products, because the basic constructs on which WCAG 2 is built do not apply to these.”

§ **role**

> text or number by which software can identify the function of a component within Web content
>
> *EXAMPLE*
>
> *Example: A number that indicates whether an image functions as a hyperlink, command button, or check box.*

§ *Applying “role” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing “web content” with “content”.

With this substitution, it would read:

role
text or number by which software can identify the function of a component within [**content**]

> Example: A number that indicates whether an image functions as a hyperlink, command button, or check box.

> NOTE (ADDED)
>
> “AccessibleRole” (or the corresponding term used in different APIs) of the Accessibility API of the platform is an example of such a role.

§ **same functionality**

> same result when used
>
> > *EXAMPLE*
> >
> > *Example: A submit "search" button on one web page and a "find" button on another web page may both have a field to enter a term and list topics in the website related to the term submitted. In this case, they would have the same functionality but would not be labeled consistently.*

§ *Applying “same functionality” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, adding a second example (and numbering the first).

With this addition, it would read:

same functionality
same result when used

> Example 1: A submit “search” button on one Web page and a “find” button on another Web page may both have a field to enter a term and list topics in the website related to the term submitted. In this case, they would have the same functionality but would not be labeled consistently.

> Example 2: A ribbon icon that saves the document that looks like an arrow pointing into a folder in one case, and an arrow pointing into a hard drive in another. In this case as well, they would have the same functionality but would not be labeled consistently.

§ **satisfies a success criterion**

> the success criterion does not evaluate to 'false' when applied to the page

§ *Applying “satisfies a success criterion” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing “page” with “non-web document or software”.

With this substitution, it would read:

satisfies a success criterion
: the success criterion does not evaluate to 'false' when applied to the [**non-web document** or **software**]

§ **set of web pages**

collection of web pages that share a common purpose and that are created by the same author, group or organization

EXAMPLE

Example: Examples include:

- *a publication which is split across multiple web pages, where each page contains one chapter or other significant section of the work. The publication is logically a single contiguous unit, and contains navigation features that enable access to the full set of pages.*
- *an e-commerce website shows products in a set of web pages that all share the same navigation and identification. However, when progressing to the checkout process, the template changes; the navigation and other elements are removed, so the pages in that process are functionally and visually different. The checkout pages are not part of the set of product pages.*
- *a blog on a sub-domain (e.g. blog.example.com) which has a different navigation and is authored by a distinct set of people from the pages on the primary domain (example.com).*

NOTE

Different language versions would be considered different sets of web pages.

§ *Applying "set of web pages" to Non-Web Documents and Software*

NOTE 1 (ADDED)

See the guidance on set of documents and set of software programs in the Key Terms section.

> NOTE 2 (ADDED)
>
> For success criteria that use the term “set of web pages”, the term is replaced by "set of non-web documents" and "set of software programs" when applying this to non-web technologies.

§ **structure**

> - The way the parts of a web page are organized in relation to each other; and
> - The way a collection of web pages is organized

§ *Applying “structure” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing “a web page” with “non-web documents or software” and replacing “collection of web pages” with “set of documents or set of software programs”.

With these substitutions, it would read:

structure

1. The way the parts of **[non-web documents** or **software]** are organized in relation to each other; and
2. The way a **[set of documents** or **set of software programs]** is organized

> NOTE (ADDED)
>
> See the guidance on sets of documents and sets of software programs in the Key Terms section.

§ **style property**

> property whose value determines the presentation (e.g. font, color, size, location, padding, volume, synthesized speech prosody) of content elements as they are rendered (e.g. onscreen, via loudspeaker, via braille display) by user agents
>
> Style properties can have several origins:
>
> - User agent default styles: The default style property values applied in the absence of any author or user styles. Some web content technologies specify a default rendering, others do not;
> - Author styles: Style property values that are set by the author as part of the content (e.g. in-line styles, author style sheets);
> - User styles: Style property values that are set by the user (e.g. via user agent interface settings, user style sheets)

§ *Applying “style property” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing “user agent(s)” with “user agent(s) or other platform software”, "web content" with "content", replacing "in-line styles, author style sheets" with "programmatically-set styles", and replacing "user agent interface settings" with "user agent, platform software or other software interface settings, or".

With these substitutions, it would read:

style property
: property whose value determines the presentation (e.g. font, color, size, location, padding, volume, synthesized speech prosody) of content elements as they are rendered (e.g. onscreen, via loudspeaker, via braille display) by [**user agents** or other **platform software**]

  Style properties can have several origins:

  - [User agent or other platform software] **default styles:** The default style property values applied in the absence of any author or user styles. Some [content] technologies specify a default rendering, others do not;

- **Author styles:** Style property values that are set by the author as part of the content (e.g. [programmatically-set styles]);
- **User styles:** Style property values that are set by the user (e.g. via [user agent, platform software or other **software** interface settings, or] user style sheets)

§ **target**

> region of the display that will accept a pointer action, such as the interactive area of a user interface component
>
> *NOTE*
>
> *If two or more targets are overlapping, the overlapping area should not be included in the measurement of the target size, except when the overlapping targets perform the same action or open the same page.*

§ *Applying "target" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "page" with "content".

With this substitution, it would read:

target
region of the display that will accept a pointer action, such as the interactive area of a user interface component

> NOTE
>
> If two or more targets are overlapping, the overlapping area should not be included in the measurement of the target size, except when the overlapping targets perform the same action or open the same [**content**].

§ **technology**

> mechanism for encoding instructions to be rendered, played or executed by user agents
>
> *NOTE 1*
>
> *As used in these guidelines "web technology" and the word "technology" (when used alone) both refer to web content technologies.*
>
> *NOTE 2*
>
> *Web content technologies may include markup languages, data formats, or programming languages that authors may use alone or in combination to create end-user experiences that range from static web pages to synchronized media presentations to dynamic Web applications.*
>
> *EXAMPLE*
>
> *Example: Some common examples of web content technologies include HTML, CSS, SVG, PNG, PDF, Flash, and JavaScript.*

§ *Applying "technology" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "web content" with "non-web document or software", "user agents" with "user agents or other software", removing the notes, and replacing the example with "Example: Some common examples of non-web document and software technologies include ODF, OOXML, Java, and C++."

With these substitutions, it would read:

technology ([non-web document or software])
mechanism for encoding instructions to be rendered, played or executed by [**user agents** or other **software**].

> Example: Some common examples of [**non-web document** and software technologies include ODF, OOXML, Java, and C++].

§ **up-event**

> platform event that occurs when the trigger stimulus of a pointer is released
>
> The up-event may have different names on different platforms, such as "touchend" or "mouseup".

§ *Applying "up-event" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary.

> NOTE (ADDED)
>
> The up-event may have different names on different platforms, such as ["PointerReleased" or "mouseup"].

§ **user agent**

> any software that retrieves and presents web content for users
>
> *EXAMPLE*
>
> *Example: Web browsers, media players, plug-ins, and other programs — including assistive technologies — that help in retrieving, rendering, and interacting with web content.*

§ *Applying “user agent” to Non-Web Documents and Software*

See the guidance on user agent in the Key Terms section.

§ **user interface component**

a part of the content that is perceived by users as a single control for a distinct function

NOTE 1

Multiple user interface components may be implemented as a single programmatic element. "Components" here is not tied to programming techniques, but rather to what the user perceives as separate controls.

NOTE 2

User interface components include form elements and links as well as components generated by scripts.

NOTE 3

What is meant by "component" or "user interface component" here is also sometimes called "user interface element".

EXAMPLE

Example: An applet has a "control" that can be used to move through content by line or page or random access. Since each of these would need to have a name and be settable independently, they would each be a "user interface component."

§ *Applying “user interface component” to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing the example with “Example: A software program has 2 controls: a text field for entering a file name and a drop down list box for choosing a folder. Each is a user interface component with a name that is settable by the software.”

With this substitution, it would read:

user interface component
a part of the content that is perceived by users as a single control for a distinct function

NOTE 1

Multiple user interface components may be implemented as a single programmatic element. "Components" here is not tied to programming techniques, but rather to what the user perceives as separate controls.

NOTE 2

User interface components include form elements and links as well as components generated by scripts.

NOTE 3

What is meant by "component" or "user interface component" here is also sometimes called "user interface element".

Example: A software program has 2 controls: a text field for entering a file name and a drop down list box for choosing a folder. Each is a user interface component with a name that is settable by the software.

§ **viewport**

> object in which the user agent presents content
>
> > *NOTE 1*
> >
> > *The user agent presents content through one or more viewports. Viewports include windows, frames, loudspeakers, and virtual magnifying glasses. A viewport may contain another viewport (e.g., nested frames). Interface components created by the user agent such as prompts, menus, and alerts are not viewports.*
>
> > *NOTE 2*
> >
> > *This definition is based on User Agent Accessibility Guidelines 1.0 Glossary [UAAG10].*

§ *Applying "viewport" to Non-Web Documents and Software*

This applies directly as written and as described in the WCAG 2 glossary, replacing "user agent" with "software".

With this substitution, it would read:

viewport
object in which the [**software**] presents content

> NOTE 1
>
> The [software] presents content through one or more viewports. Viewports include windows, frames, loudspeakers, and virtual magnifying glasses. A viewport may contain another viewport (e.g., nested frames). Interface components created by the [software] such as prompts, menus, and alerts are not viewports.

> NOTE 2
>
> This definition is based on User Agent Accessibility Guidelines 1.0 Glossary [UAAG10].

§ **web page**

a non-embedded resource obtained from a single URI using HTTP plus any other resources that are used in the rendering or intended to be rendered together with it by a user agent

NOTE 1

Although any "other resources" would be rendered together with the primary resource, they would not necessarily be rendered simultaneously with each other.

NOTE 2

For the purposes of conformance with these guidelines, a resource must be "non-embedded" within the scope of conformance to be considered a web page.

EXAMPLE 1

Example 1: A web resource including all embedded images and media.

EXAMPLE 2

Example 2: A web mail program built using Asynchronous JavaScript and XML (AJAX). The program lives entirely at http://example.com/mail, but includes an inbox, a contacts area and a calendar. Links or buttons are provided that cause the inbox, contacts, or calendar to display, but do not change the URI of the page as a whole.

EXAMPLE 3

Example 3: A customizable portal site, where users can choose content to display from a set of different content modules.

EXAMPLE 4

Example 4: When you enter "http://shopping.example.com/" in your browser, you enter a movie-like interactive shopping environment where you visually move around in a store dragging products off of the shelves around you and into a visual shopping cart in front of you. Clicking on a product causes it to be demonstrated with a specification sheet floating alongside. This might be a single-page website or just one page within a website.

Applying "Web Page" to Non-Web Documents and Software

This applies directly as written and as described in the WCAG 2 glossary.

NOTE (ADDED)

For those success criteria that use the term "web page", WCAG2ICT provides specific replacement term(s) for "Web page".

Privacy Considerations

This section is non-normative.

This Working Group Note does not introduce any new privacy considerations. However, when implementing WCAG 2 success criteria in the context of non-web ICT, information about a user's accessibility needs or preferences might be exposed; a user could be harmed by disclosure and misuse of that information. It is best practice to choose implementations that reduce the potential for fingerprinting or other identification and tracking of users, and that the only data collected is data necessary to enable the accessibility features.

NOTE

The WCAG 2 Privacy Considerations section also notes specific success criteria that are identified to have possible implications for privacy that could also exist for non-web ICT.

Security Considerations

This section is non-normative.

This Working Group Note does not introduce any new security considerations. Since any software feature has the potential to compromise security, take care when implementing non-web ICT features added to meet WCAG 2 success criteria.

> NOTE
>
> The WCAG 2 Security Considerations section lists specific success criteria with possible implications for security, which could also exist for non-web ICT.

A. Success Criteria Problematic for Closed Functionality

There are success criteria that can be problematic for developers of ICT with closed functionality. Some criteria discuss making information available in text (which can be read by assistive technologies), making it “programmatically determinable” (rendered by a user agent and readable by assistive technologies), or doing something else to make content compatible with assistive technologies. Where ICT with closed functionality doesn’t support use of assistive technology or the platform does not have an accessibility API, providing equivalent information and operation through another mechanism, such as functions built into the software that behave like assistive technology, would help meet the intent of these success criteria. See also the Comments on Closed Functionality section.

Other success criteria would apply to ICT with closed functionality either if they are partially closed or if they allow for the connection of some types of devices. As an example, Success Criterion 2.1.1 Keyboard would apply to ICT that is closed to screen readers, but have a physical keyboard, a connector for standard keyboards, or allow the installation of alternate keyboards. While these criteria, as written, are not always applicable to ICT with closed functionality, most of them can inform and aid development of built-in features needed to make ICT with closed functionality accessible.

For non-web software on ICT with closed functionality, those who implement this document (WCAG2ICT) should consider the applicability of individual WCAG 2 success criteria on a criterion-

by-criterion basis. Alternate accessibility provisions might be needed to cover the user needs addressed by the following success criteria:

- 1.1.1 Non-text Content — Depends upon text (or a text alternative) being in a programmatically determinable form.
- 1.2.1 Pre-recorded video — One of the options available to authors for Success Criterion 1.2.1 is providing a media alternative that is text which, in the absence of connected assistive technology, would need to be made available in different modalities.
- 1.2.3 Audio description or Media Alternative — One of the options available to authors for Success Criterion 1.2.3 is providing a media alternative that is text which, in the absence of connected assistive technology, would need to be made available in different modalities.
- 1.3.1 Info and Relationships — Requires that information be in a programmatically determinable form or in text (that is programmatically determinable).
- 1.3.2 Meaningful Sequence — Requires information (i.e, a correct reading sequence) in a programmatically determinable form. An equivalent for ICT with closed functionality would be to provide a meaningful reading sequence through auditory output or some other non-visual means that helps users correlate the output with the corresponding information displayed on the screen.
- 1.3.4 Orientation — ICT with closed functionality that has fixed-in-place displays or other limitations that prevent modifying the physical display orientation should be considered as examples that are covered under the essential exception. See the note in the section Applying SC 1.3.4 Orientation to Non-Web Software.
- 1.3.5 Identify Input Purpose — Depends upon information in a programmatically determinable form; in the absence of programmatic capabilities, text labels need to be specific and be provided to the user in other modalities (e.g. auditory).
- 1.4.2 Audio Control — The intent of this success criterion is to avoid interference of audio with assistive products, which are not available in ICT with closed functionality. If the built-in accessibility features of the ICT with closed functionality provide speech output, then the interference may happen and this success criterion applies. In addition, there are existing requirements in regulations (e.g., the EN 301 549 and U.S. Revised 508 Standards) that address volume control for ICT with closed functionality.
- 1.4.3 Contrast (Minimum) — There are cases where applying this success criterion to non-web software on ICT with closed functionality is problematic:
  - When the contrast of the content is determined by the hardware and not modifiable by the software author, it may not be possible to meet this success criterion.

> NOTE 1
> Contrast requirements for hardware are out of scope for WCAG2ICT (and this success criterion).

- When the color contrast ratio cannot be programmatically measured due to system limitations (e.g. lockdown), precise quantifiable testing of color contrast cannot be performed by a third party. In such cases, the software author would need to confirm that the color combinations used meet the contrast requirement.

> NOTE 2
> Photographs (e.g., of a hardware display) are not sufficient for testing that content meets this success criterion. This is because the quality of the lighting, camera, and physical aspects of the hardware display can dramatically affect the ability to capture the content for testing purposes.

- 1.4.4 Resize Text — Non-web software on ICT with closed functionality may offer more limited text rendering support than the support found in user agents for the Web. As a result, meeting Success Criterion 1.4.4 in a closed environment may place a much heavier burden on the content author.
- 1.4.5 Images of Text — To enable assistive technology to modify displayed text (e.g., adjusting contrast, increasing font size), machine-readable text is needed, as opposed to mere images of text. Not all ICT with closed functionality has the capability to support visual modification of displayed text or images of text, given there is no interoperability with assistive technology and/or lack of platform support.
- 1.4.10 Reflow — Some non-web software on ICT with closed functionality does not support scrolling content, or zooming, or changing the size of a viewport or scrollable content area to the specified width/height (examples include, but are not limited to, software for self-service transaction machines or kiosks). Therefore, some other non-WCAG requirements would be needed for ICT with closed functionality to ensure that content is readable by persons with low vision without scrolling in two dimensions.

> NOTE 3
> Some ICT with closed functionality does not display large chunks of text and only has UI controls. In such cases, two-dimensional scrolling to access the text and UI controls may be considered essential, thus meeting an exception, and the success criterion would be satisfied.

- 1.4.11 Non-text Contrast — There are cases where applying this success criterion to non-web software on ICT with closed functionality is problematic:

- When the contrast of the content is determined by the hardware and not modifiable by the software author, it may not be possible to meet this success criterion.

  > NOTE 4
  > Contrast requirements for hardware are out of scope for WCAG2ICT (and this success criterion).

- When the color contrast ratio cannot be programmatically measured due to system limitations (e.g. lockdown), precise quantifiable testing of color contrast cannot be performed by a third party. In such cases, the software author would need to confirm that the color combinations used meet the contrast requirement.

  > NOTE 5
  > Photographs are not sufficient for testing that content meets this success criterion. This is because the quality of the lighting, camera, and physical aspects of the hardware display can dramatically affect the ability to capture the content for testing purposes.

- 1.4.12 Text Spacing — In non-web software on ICT with closed functionality the ability for users to modify line, paragraph, letter, or word spacing is rarely supported. Regardless, the success criterion applies as written and as noted in the Applying SC 1.4.12 Text Spacing to Non-Web Documents and Software.
- 2.1.1 Keyboard — Assumes operation via a keyboard interface which also allows for alternative input devices. It may not be possible to satisfy this success criterion when the ICT does not have a built-in keyboard, and it also does not support an alternative input method (hardware or software) that provides keyboard-like access to all functionality.

  > NOTE 6
  > A keypad that provides full access to functionality might be considered a keyboard.

- 2.1.2 No Keyboard Trap — This criterion applies when focus can be moved using a keyboard interface. In some ICT with closed functionality, tactile input like numeric keypads or other functional groups of keys may be available, but there is no mechanism for onscreen focus; for example, the keys are mapped directly to functions without moving focus between on-screen controls. In this case, there is no concept of focus, and therefore keyboard traps cannot exist and this success criterion would be satisfied.
- 2.1.4 Character Key Shortcuts — ICT with closed functionality might lack a mechanism for keyboard shortcuts because their mode of operation revolves around a single key performing a single function. For such systems, this success criterion is satisfied.

- 2.4.1 Bypass Blocks — The WCAG2ICT interpretation of this success criterion replaces "sets of Web pages" with "sets of software programs" which are extremely rare - especially for non-web software on ICT with closed functionality. However, being able to bypass blocks of content that are repeated within software is generally considered best practice.
- 2.4.4 Link Purpose (In Context) — This success criterion relies upon text and context being made available in a programmatically determinable form.
- 2.4.5 Multiple Ways — The WCAG2ICT interpretation of this success criterion replaces "set of Web pages" with "set of software programs". Such sets, particularly in the context of non-web software on ICT with closed functionality, are exceedingly rare. There are a number of notes in the section Applying SC 2.4.5 Multiple Ways to Non-Web Documents and Software that are applicable to non-web software on ICT with closed functionality.
- 2.4.7 Focus Visible — Presumes that there is a mode of operation where focus can be moved and controlled by keyboard. Some ICT with closed functionality may offer tactilely discernible input such as a numeric keypad or other functional groups of keys, but do not offer any mechanism for conveying focus because the user interface is designed not to need that. For example, the keys are used to select options from a spoken menu rather than to move an onscreen focus element between multiple options. In this case, there is no concept of focus, thus there is no need for a visible indicator and this success criterion would be satisfied.
- 2.5.2 Pointer Cancellation — As noted in the section Applying SC 2.5.2 Pointer Cancellation to Non-Web Documents and Software, examples of ‘essential’ functionality are features for meeting environmental energy usage requirements (like waking a device from sleep, power saver mode, and low power state).
- 2.5.3 Label in Name — Requires information in a programmatically determinable form; specifically, the programmatic name contains the text of the visual label.
- 2.5.8 Target Size (Minimum) — This success criterion uses CSS pixels for defining the target size. ICT with closed functionality may not use CSS pixels as a standard measurement, but the definition of ‘CSS pixel’ still applies as described in Applying “CSS pixel” to Non-Web Documents and Software. If the system supports a density-independent pixel measurement, it should be used in place of CSS pixels.

> NOTE 7
> If the viewing distance and pixel density of the system are unknown, approximating the reference pixel as described in Applying “CSS pixel” to Non-Web Documents and Software is not possible.

> NOTE 8
> For non-web software designed to run on specific known hardware, a physical size standard would be more straightforward to apply, as calculations for a CSS pixel are dependent on the viewing distance or pixel density of the display.

- 3.1.1 Language of Page — Depends upon language information being in a programmatically determinable form intended to drive correct pronunciation. Where another mechanism achieves correct pronunciation for ICT with closed functionality, such as self-voicing, the intent of this success criterion would be met.
- 3.1.2 Language of Parts — Depends upon language information in a programmatically determinable form intended to drive correct pronunciation. Where another mechanism achieves correct pronunciation for ICT with closed functionality, such as self-voicing, the intent of this success criterion would be met.
- 3.2.3 Consistent Navigation — This success criterion is interpreted to only apply to “sets of software programs” which are very rare. See the second note in the section Applying SC 3.2.3 Consistent Navigation to Non-Web Documents and Software.
- 3.2.4 Consistent Identification — This success criterion is interpreted to only apply to “sets of software programs” which are very rare. See the second note in the section Applying SC 3.2.4 Consistent Identification to Non-Web Documents and Software.
- 3.2.6 Consistent Help — The WCAG2ICT interpretation of this success criterion replaces "sets of Web pages" with "sets of software programs" which are extremely rare - especially for non-web software on ICT with closed functionality. However, providing consistent access to help is generally considered best practice.
- 3.3.1 Error Identification — Requires error information to be provided as text, noting that the WCAG definition of text is that it be "programmatically determinable".
- 3.3.8 Accessible Authentication (Minimum) — There are situations where meeting this success criterion is problematic for ICT with closed functionality:
  - Systems that are designed for shared use (such as in a public library) or have closed functionality might block mechanisms typically used to assist the user, such as copying authentication information from a password manager. Instead, an alternative authentication method might be needed, such as an identity card scanner.
  - Where standards for banking or security have authentication requirements that are regulated or strictly enforced, those requirements may be judged to take legal precedence over Success Criterion 3.3.8 Accessible Authentication (Minimum).
- 4.1.1 Parsing

- When WCAG 2.0 and 2.1 were written, the Intent of Success Criterion 4.1.1 was to provide consistency so that different user agents or assistive technologies would yield the same result.
- In WCAG 2.2, Success Criterion 4.1.1 Parsing was made obsolete and WCAG 2.2 removed it as a requirement, so this success criterion is not applicable.

- 4.1.2 Name, Role, Value — Requires information in a programmatically determinable form.
- 4.1.3 Status Messages — Depends upon status messages being programmatically determinable using role or properties.

> NOTE 9
> Non-web software on ICT with closed functionality would need equivalent facilitation to provide access to status messages.

B. Background on Text / Command-Line / Terminal Applications and Interfaces

B.1 How Text Interfaces Are Realized

The interface of a text application is realized through a server application directing which characters should be placed on the screen, along with either a hardware terminal or a terminal application that displays the characters. The client terminal application for text applications is analogous to a web user agent for web pages. Also, like web applications, text applications may execute primarily on a remote server or execute locally.

Some text applications render like a TeleTYpewriter (TTY); their output is always appended, like an ever-growing file. Such text applications are often called “command-line applications” or occasionally “TTY-applications”, and their output can optionally be redirected to a file for later review. Others explicitly place text into a matrix of fixed width character cells on a screen (sometimes with specific foreground and background colors).

Historically, input to the text application itself is provided exclusively through a keyboard interface, though Automatic Speech Recognition (ASR) based voice input is sometimes now an alternative option - especially on mobile devices.

§ B.2 How Text Applications Have Been Made Accessible Via Assistive Technology

Strategies for making text applications accessible through assistive technology involve two key tasks: (1) obtaining all of the text displayed in the interface, and (2) performing an analysis on that text to detect screen updates and attempt to discern structural elements.

For example, a text application screen reader might directly access the matrix of character cells in the interface and provide a screen review mechanism for the user to review that matrix of characters (by sending the output to synthetic speech and/or a braille display). Alternately, a text application screen reader might directly consume the output rendered (perhaps by acting as its own terminal application or by analyzing the “TTY” output). A text application screen reader might also attempt to analyze the spacing and layout of the text in the matrix, to provide features such as reading columns of text in a multi-column layout; discerning headers through analysis of line spacing, indentation, and capitalization; and discerning input fields or user interface components by scanning for the use of inverse video, for text appearing in brackets, or for text from the character graphics codepage (ASCII codes greater than ‘0x7F’). Some of this analysis might also be done through the use of filter tools that transform the output of a program (e.g., through reformatting “TTY” output rendered to a file or as direct input to a filter tool).

Similarly, a text application screen magnifier would gain access to the matrix of character cells to magnify them or re-display them in a larger font. It would scan for screen refreshes and updates and then apply heuristics to what had changed in order to decide what sub-matrix of character cells should appear in a magnified view. It would also scan for inverse video and a moving text cursor to track text being input by the user (and might combine the text matrix scanning with scanning of the keyboard input to match user input to what is appearing on the screen).

§ B.3 Applying WCAG 2 to Text Applications

To apply WCAG to text applications, it is necessary to apply the glossary terms accessibility supported and programmatically determined in the context of how text applications are rendered and the history of assistive technologies that made them accessible.

As noted above, in a text interface the terminal application renders the characters on the screen, just as a web browser typically renders content for a web application. As an example, for success criterion 1.4.4 Resize Text, a text application could achieve 200 percent resizing when the terminal application client that is rendering it has this capability (cf. WCAG 2 Technique G142 Using a technology that has

commonly-available user agents that support zoom). Many web pages and web applications use this approach to meet success criterion 1.4.4 Resize Text through no explicit action of their own.

A similar approach could also be used for success criterion 1.4.3 Contrast (minimum) (cf. WCAG 2 Technique G148: Not specifying background color, not specifying text color, and not using technology features that change those defaults): relying on the terminal application client to render the text with sufficient contrast against the background. In fact, many terminal applications allow the user to force all text to share a single user-chosen foreground color (and a single user-chosen background color), overriding the text application's specified colors to meet the user's desires or needs.

Since many assistive technology analysis techniques depend upon discerning the location of the text input cursor, terminal application use of “soft cursors” and “highlight bars” may bypass those analysis techniques and cause failures of success criteria.

> NOTE 1
> It is outside of the scope of this document to define WCAG techniques for non-web ICT. These examples are simply illustrations of how WCAG 2 success criteria can be applied to this class of non-web software applications.

The way to think about "accessibility supported" and "programmatically determined" may seem a little different for text applications, but the definitions are unchanged. Unlike the semantic objects of graphical user interfaces and web pages, the output of text-based applications consists of plain text. A terminal emulator acts as the user agent for text-based applications; it might render some content such as escape codes as semantic elements, but otherwise exposes only lines of text to assistive technology. Where assistive technology is able to interpret the text and any semantic objects accurately, the content is "programmatically determinable"—even though no explicit markup was necessarily used to make it so.

> NOTE 2
> The terminal application itself is “traditional” non-web software ICT. It is only for the text application that there is a need to take this approach with these glossary terms.

§ C. Acknowledgements

§

C.1 Active Participants of the WCAG2ICT Task Force Involved in the Development of This Document

- Shadi Abou-Zahra (Amazon)
- Bruce Bailey (Invited Expert)
- Phil Day (NCR Atleos)
- Tamsin Ewing (W3C Staff)
- Loïc Martínez Normand (Universidad Politécnica de Madrid)
- Laura Miller (US General Services Administration)
- Daniel Montalvo (W3C Staff)
- Mary Jo Mueller (IBM Corporation)
- Mike Pluke (Invited Expert)
- Gregg Vanderheiden (W3C Invited Expert)

§ C.2 Participants in the AG Working Group that Actively Reviewed and Contributed

- Jonathan Avila (Level Access)
- Daniel Bjorge (Deque Systems, Inc.)
- Alastair Campbell (Nomensa)
- Laura Carlson (Invited Expert)
- DJ Chase (Invited Expert)
- Azlan Cuttilan (Invited Expert)
- Jennifer Delisi (Invited Expert)
- Steve Faulkner (TetraLogical Services Ltd.)
- Wilco Fiers (Deque Systems, Inc.)
- Detlev Fischer (Invited Expert)
- Mike Gifford (Invited Expert)
- Michael Gower (IBM Corporation)
- Jan Jaap de Groot (Abra)

- Shawn Henry (W3C Staff)
- Andrew Kirkpatrick (Evinced Inc.)
- Patrick Lauke (TetraLogical)
- Todd Libby (Invited Expert)
- David MacDonald (Invited Expert)
- Rachael Bradley Montgomery (Library of Congress)
- Lori Oakley (Oracle Corporation)
- Scott O'Hara (Microsoft Corporation)
- Kimberly Patch (Invited Expert)
- Melanie Philipp (Invited Expert)
- Julie Rawe (Understood)
- Roberto Scano (Invited Expert)
- Lisa Seeman-Horwitz (Invited Expert)
- Poornima Badhan Subramanian (Invited Expert)
- Ben Tillyer (University of Oxford)
- Kevin White (W3C Staff)
- Frankie Wolf (Invited Expert)

WCAG2ICT Task Force participants are also in the AG working group, but are not repeated here.

§ C.3 Other Participants of the WCAG2ICT Task Force

- Charles Adams (Oracle Corporation)
- Fernanda Bonnin (Microsoft Corporation)
- Devanshu Chandra (Deque Systems, Inc.)
- Mitchell Evan (TPGi)
- Thorsten Katzmann (IBM Corporation)
- Sam Ogami (Invited Expert)
- Jennifer Strickland (MITRE Corporation)

- Shawn Thompson (Shared Services Canada)
- Bryan Trogdon (Google LLC)

§ C.4 Previous Contributors

The following people contributed to the development of the WCAG2ICT Note:

Charles Adams, Matthew Atkinson, Fernanda Bonnin, Judy Brewer, Devanshu Chandra, Michael Cooper, Mitchell Evan, Anastasia Lanz, Janina Sajka, Olivia Hogan-Stark, Thorsten Katzmann, Chris Loiselle, Sam Ogami, Shawn Thompson, Jason White

§ C.5 Enabling Funders

This publication has been funded in part with U.S. Federal funds from the Health and Human Services, National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR), initially under contract number ED-OSE-10-C-0067 and now under HHS75P00120P00168. The content of this publication does not necessarily reflect the views or policies of the U.S. Department of Health and Human Services or the U.S. Department of Education, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. Government.

Work on this publication was part of the WAI-CooP Project, a European Commission (EC) co-funded project, Horizon 2020 Program (101004794).

§ D. Changelog

§ D.1 Changes since the 21 August 2025 Note

- 2025-12-04 sotd updates
- 2025-10-30 2.4.2 Page Titled - Not all non-web docs and SW have titles that describe the topic or purpose

- 2025-10-30 2.4.4 Link Purpose: Adjust "link" note to apply to non-web docs as well
- 2025-10-24 1.3.4 Orientation: update guidance for non-web documents and software and add examples
- 2025-10-09 1.4.4 Resize Text: Remove note regarding minimum text size
- 2025-09-18 1.4.13 Content on Hover or Focus: additional word substitutions
- 2025-09-16 1.4.4 Resize text: updates to resolve issue 772 and incorporate EN note
- 2025-10-22 2.3.1 Three Flashes or Below Threshold: Add 2 new notes
- 2025-10-10 1.2.3 and 1.2.5: Add note describing audio descriptions
- 2025-09-26 1.4.13 Content on Hover or Focus: Update note for "links" word replacement
- 2025-09-11 2.4.4 Link Purpose: Update Note 1 to simplify and remove "outside a user interface control,"

§ D.2 Changes since the 15 November 2024 Note

- 2025-07-31 Add note describing device-independent keyboard interface services
- 2025-08-19 1.4.10 Reflow: Remove Note 5 from non-web documents
- 2025-08-12 2.1.1 Keyboard - Reinstate interpretation for non-web documents
- 2025-08-12 1.4.4 Resize text - add note from EN 301 549
- 2025-07-31 Make adjustments to 1.4.1 where platforms provide text scaling for all text, but not to 200% for some text
- 2025-04-25 Update note 1 of 2.1.1 Keyboard guidance to make it more understandable
- 2025-07-17 2.1.1 Keyboard - Updates to notes prompted by the EN 301 549
- 2025-07-17 3.1.2 Language of Parts - Add modified EN 301 549 note
- 2025-06-19 Simplify word replacement in "target" definition to use "content" as word replacement for "page"
- 2025-06-12 More consistency changes for note used in 1.4.2, 2.1.2, 2.2.2, and 2.3.1
- 2025-06-21 Update 4.1.2 to have a word replacement for "user agent"
- 2025-06-16 Adjust note in 1.4.2, 2.1.2, 2.2.2, and 2.3.1 to be consistent verbiage and avoid use of "must"
- 2025-04-29 Add Notes 2 and 3 to key term "content"

- 2025-06-20 SC 2.2.2: Update Note 5 to give full understanding doc name and link and not use "informative" which has a specific meaning in WCAG
- 2025-06-27 1.4.10: Add note for non-web doc from first 2 paragraphs of non-web software note
- 2025-07-03 1.3.1: Add note for non-web documents from the EN 301 549

E. References

E.1 Informative references

[etsi-en-301-549]
ETSI EN 301 549 V3.2.1 (2021-03): Accessibility requirements for ICT products and services. ETSI. March 2021. Published. URL: http://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf

[ISO_9241-171]
Ergonomics of human-system interaction Part 171: Guidance on software accessibility. International Standards Organization. 2008. URL: https://www.iso.org/standard/39080.html

[ISO/IEC_13066-1]
Information technology - Interoperability with assistive technology (AT) Part 1: Requirements and recommendations for interoperability. International Standards Organization. 2011. URL: https://www.iso.org/standard/53770.html

[UNDERSTANDING-WCAG22]
Understanding Web Content Accessibility Guidelines 2.2. URL: https://www.w3.org/WAI/WCAG22/Understanding/

[WCAG20]
Web Content Accessibility Guidelines (WCAG) 2.0. Ben Caldwell; Michael Cooper; Loretta Guarino Reid; Gregg Vanderheiden et al. W3C. 11 December 2008. W3C Recommendation. URL: https://www.w3.org/TR/WCAG20/

[WCAG21]
Web Content Accessibility Guidelines (WCAG) 2.1. Michael Cooper; Andrew Kirkpatrick; Joshue O'Connor; Alastair Campbell. W3C. 6 May 2025. W3C Recommendation. URL: https://www.w3.org/TR/WCAG21/

[WCAG22]

Web Content Accessibility Guidelines (WCAG) 2.2. Michael Cooper; Andrew Kirkpatrick; Alastair Campbell; Rachael Bradley Montgomery; Charles Adams. W3C. 12 December 2024. W3C Recommendation. URL: https://www.w3.org/TR/WCAG22/

[WCAG22-TECHS]

Techniques for WCAG 2.2. URL: https://www.w3.org/WAI/WCAG22/Techniques/