CLS – UI COMPONENTS

STRATEGY FOR

KEYBOARD ACCESSIBLITY

PREPARED BY
ROSSANA BABAKHANI
UX RESEARCHER & DESIGNER

01
PURPOSE

Keyboard interaction is an important part of how users work across Clinisys applications. For some users it is an accessibility requirement; for others, particularly frequent and expert users, it is the fastest way to complete repetitive tasks.

The purpose of this strategy is to establish a consistent approach to keyboard interaction across Clinisys products. Users should be able to navigate, understand, and complete a workflow without relying on a mouse, while maintaining the speed and shortcuts that are important to existing workflows.

The strategy focuses on four areas:

- **Navigation** — users can move through the interface in a logical and predictable way.
- **Focus** — users can always identify where they are and where an action will occur.
- **Interaction** — components have keyboard behavior appropriate to the task.
- **Efficiency** — shortcuts support frequent and repetitive workflows without becoming the only way to complete an action.

02
NAVIGATION

Keyboard navigation should follow the interface's structure and workflow.

Tab and **Shift + Tab** move between interactive elements in a logical order. Users should not be required to tab through non-interactive content or through every data cell simply to move to the next part of a screen.

Where a page contains repeated navigation or large sections of content, provide a way to skip directly to the main content or another key section.

Navigation patterns should remain consistent across Clinisys applications so that users do not have to relearn basic keyboard behavior from one workflow to another.

03
FOCUS

Keyboard focus must always be visible.

The focused element should have a clear visual indicator that is distinguishable from hover, selection, error, and edit states.

Focus should also behave predictably when the interface changes. For example, after closing a dialog, focus should return to the element that opened it or another logical point in the workflow.

When content is dynamically added, removed, refreshed, or updated, focus should not move unexpectedly.

04
COMPONENT INTERACTION

Keyboard behavior should match the component type and the task it supports.

Standard controls should follow established keyboard conventions wherever possible. Buttons, links, menus, dialogs, forms, tables, and other common components should not introduce custom keyboard behavior when an established pattern already exists.

More complex components may require their own internal navigation model.

A Data Grid, for example, may use arrow keys to move between cells because users are working within the dataset. A Table may use a simpler navigation model because users primarily review and act on records.

This distinction is important. Similar-looking components should not automatically be given the same keyboard behavior when the user interaction is fundamentally different.

05
KEYBOARD SHORTCUTS

Keyboard shortcuts should improve efficiency without being required to use the product.

Shortcuts are particularly valuable for frequent actions and workflows involving repeated input. They should be used deliberately rather than adding shortcuts for every available action.

Shortcuts should be:

- consistent across applications where the action is the same;
- contextual where the same key has different meanings in different components;
- designed to avoid conflicts with browser, operating system, assistive technology, and dictation commands;
- available alongside another accessible method of performing the same action.

Existing shortcuts that are important to established workflows should be evaluated before they are changed or removed.

06
SHORTCUT DISCOVERABILITY

Users should not be expected to memorize or already know available keyboard shortcuts.

Shortcuts should be discoverable through the interface. Depending on the context, this may include shortcut hints alongside actions, tooltips, contextual help, or a keyboard shortcut reference.

The information shown should be relevant to the current workflow or component rather than presenting users with an unnecessarily long list of commands.

This is particularly important for components such as Data Grid, where the keyboard interaction model can be considerably more extensive.

07
FORMS AND DATA ENTRY

Forms should support efficient keyboard entry from beginning to end.

Tab order should follow the expected workflow, and users should be able to move between fields without unnecessary changes in focus.

Validation should identify the field containing the problem and provide a keyboard-accessible way to reach and correct errors.

Repeated data-entry workflows should be reviewed specifically for unnecessary keystrokes. Actions that are acceptable when performed once can become significant barriers when repeated dozens of times within a workflow.

08
COMPLEX COMPONENTS

Components such as Data Grids, menus, dialogs, autocomplete controls, hierarchical views, and other composite components require defined keyboard interaction patterns.

For each complex component, the specification should define:

- how focus enters and leaves the component;
- how users navigate within it;
- how selection works;
- how actions are activated;
- how editing begins and ends, where applicable;

- how users cancel or return to the previous state;
- what happens to focus after an action is completed.

These behaviors should be part of the component specification rather than determined independently by each product team.

09
FOCUS, SELECTION, AND EDIT STATE

Focus, selection, and editing are different interaction states and should not be treated as interchangeable.

- **Focus** indicates to the user where keyboard interaction is currently.
- **Selection** indicates which item or data will be affected by an action.
- **Edit state** tells the user that input will modify a value.

Where a component supports more than one of these states, each should have a clear and distinguishable visual treatment.

This is particularly important for Data Grids and other components where a user can navigate to an item without immediately editing or selecting it.

10
DICTATION AND ASSISTIVE TECHNOLOGY

Keyboard interaction should work alongside assistive technologies and dictation software.

Shortcuts should be evaluated for conflicts with commonly used browser, operating system, assistive technology, and dictation commands.

Where keyboard commands are documented, they should be written in a way that also supports users issuing keyboard commands through dictation software.

Keyboard accessibility should complement assistive technology rather than assume that users interact exclusively through a physical keyboard.

10
TESTING AND VALIDATION

Keyboard accessibility should be tested as part of component and workflow design, not only during final accessibility review.

Testing should include:

- completing the workflow without a mouse;
- checking logical Tab and Shift + Tab order;
- verifying visible focus throughout the workflow;
- entering and leaving complex components;
- testing selection and edit states;
- opening and closing dialogs and menus;
- recovering from validation errors;
- confirming that focus remains predictable after dynamic updates;
- testing documented keyboard shortcuts.

High-frequency workflows should also be evaluated for efficiency. The number of keystrokes and the frequency of mode changes required to complete a task should be considered alongside technical accessibility compliance.

GUIDING PRINCIPLE

A keyboard-accessible interface is not simply one where every control can eventually be reached with the Tab key.

Users need to know where they are, what they can do, what their action will affect, and how to move efficiently through the workflow.

The goal is consistent, predictable keyboard interaction across Clinisys while allowing components such as Tables and Data Grids to use the navigation model appropriate to the work being performed.