Contents

1 ABOUT THIS DOCUMENT .............................................................. 5
  1.1 Audience ..................................................................................... 5

2 INTRODUCTION .......................................................................... 6
  2.1 How to Read this Document .......................................................... 6

3 THE DYALOG DEVELOPMENT ENVIRONMENT ............................ 7
  3.1 Menu Bar ....................................................................................... 7
    3.1.1 Dyalog Menu .......................................................................... 7
    3.1.2 File Menu ................................................................................ 7
    3.1.3 Edit Menu ............................................................................... 8
    3.1.4 View Menu ............................................................................. 8
    3.1.5 Window Menu ......................................................................... 9
    3.1.6 Action Menu .......................................................................... 10
    3.1.7 Threads Menu .......................................................................... 10
    3.1.8 Help Menu ........................................................................... 11
  3.2 Keyboard Key Mappings for APL Glyphs ................................. 11
  3.3 Session User Interface ................................................................. 12
    3.3.1 Caption .................................................................................. 13
    3.3.2 Language Bar .......................................................................... 13
    3.3.3 Session Window ...................................................................... 13
    3.3.4 Status bar ............................................................................... 13
    3.3.5 Workspace Explorer ............................................................... 14
    3.3.6 Debug Information Window ..................................................... 14

4 INPUT WINDOWS ...................................................................... 16
  4.1 Session Window .......................................................................... 16
  4.2 Edit Window ................................................................................ 16
    4.2.1 Toolbar .................................................................................. 17
    4.2.2 Search and Replace ................................................................. 18
    4.2.3 Exiting the Edit Window ............................................................ 20
  4.3 Trace Window ............................................................................. 20
    4.3.1 Toolbar .................................................................................. 21
    4.3.2 Search .................................................................................... 21
    4.3.3 Exiting the Trace Window .......................................................... 22

5 WORKING IN A DYALOG SESSION ........................................ 23
  5.1 Keyboard Shortcuts and Command Codes ............................ 23
  5.2 Accented Characters ................................................................. 23
  5.3 Navigating the Windows ............................................................ 23
  5.4 Display of Windows ................................................................... 24
  5.5 Entering APL Characters ........................................................... 24
  5.6 Entering Expressions ................................................................. 25
    5.6.1 Paired Enclosures ................................................................. 25
    5.6.2 Autocomplete ........................................................................ 25
1 About This Document

This document is specific to using Dyalog on the macOS operating system. For information on the Dyalog development environment on other operating systems, consult the appropriate documents.

This document introduces the Dyalog development environment (user interface). It describes the windows, menus and customisation options that are available as well as the keycode-keystroke mappings that can be used within the different input windows.

The Dyalog Session can be extensively customised; this document assumes that the default configuration is in use.

This document is only part of the full documentation set for Dyalog (available to download from http://docs.dyalog.com). In cases where the information given is different between this document and any of the others, this document should be regarded as the definitive source for Dyalog for macOS.

1.1 Audience

It is assumed that the reader has a basic knowledge of macOS and a working knowledge of Dyalog (for information on the resources available to help develop your Dyalog knowledge, see http://www.dyalog.com/introduction.htm – new users might also find that it is helpful to select the Show tips for glyphs check box in the General tab of the Preferences dialog box, as detailed in section 7.2.1).
2 Introduction

The Dyalog development environment offers a user-friendly means of interacting with the interpreter. Support for Dyalog on macOS was new at Dyalog version 14.1, and the development environment is not yet as feature-rich as that on Microsoft Windows; however, additional functionality continues to be added.

2.1 How to Read this Document

Depending on your level of experience with Dyalog/APL and your preferred learning practices, you might want to go straight to a particular section of this document in preference to reading through it in the order in which it is presented. To assist with this, the content of this document is grouped as follows:

- Chapter 3 introduces the basic features of the user interface that will be seen on initially starting a Dyalog Session.
- Chapter 4 describes the different windows that can be accessed and the functionality that is available with each type.
- Chapter 5 contains the details of working within the development environment. This includes information on how to enter and execute expressions, along with descriptions of the functionality that is provided to simplify this.
- Chapter 6 concentrates on the language elements of Dyalog that are not supported with Dyalog for macOS.
- Chapter 7 explains how the appearance and behaviour of a Dyalog Session can be customised to meet personal preferences and corporate guidelines.
# The Dyalog Development Environment

When a Dyalog Session is started, the Dyalog development environment is displayed. This means that:

- the menu bar menu options change to those appropriate for Dyalog (see section 3.1)
- the keyboard key mappings for APL glyphs are enabled (see section 3.2)
- the Dyalog Session user interface is displayed (see section 3.3)

## 3.1 Menu Bar

The menu bar menu options when Dyalog is the active application are shown in Figure 1 – this section details each of the items in these menus.

![Figure 1. Menu bar menu options](image)

The menu names in the menu bar and the options under each menu can be customised (see section 7.2.7).

### 3.1.1 Dyalog Menu

The options available under the Dyalog menu are detailed in Table 1. These are standard application-specific menu options on macOS.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Dyalog</td>
<td>Displays the About dialog box, which provides details of the current Dyalog installation.</td>
</tr>
<tr>
<td>Preferences</td>
<td>Opens the Preferences dialog box (see section 7.2).</td>
</tr>
<tr>
<td>Quit</td>
<td>Terminates the Dyalog Session (see section 5.10).</td>
</tr>
</tbody>
</table>

### 3.1.2 File Menu

The options available under the File menu are detailed in Table 2. These control the RIDE-enabled APL process connections (both on local and remote machines).

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open…</td>
<td>Prompts for a filename. If a workspace file is selected (a .dws file) then it is loaded into the Session; any other file is opened in the Edit window.</td>
</tr>
<tr>
<td>New Session</td>
<td>Starts a new Dyalog Session (a new instance of the interpreter).</td>
</tr>
</tbody>
</table>
Connect… | Opens the **RIDE-Dyalog Session** dialog box, enabling you to connect to specified or remote RIDE-enabled Dyalog process. For information on RIDE, connecting to remote interpreters and the **RIDE-Dyalog Session** dialog box, see the **RIDE User Guide**.

### 3.1.3 Edit Menu

The options available under the **Edit** menu are detailed in Table 3. These assist with manipulating text within (and between) windows.

**Table 3. Edit menu options**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>Reverses the previous action</td>
</tr>
<tr>
<td>Redo</td>
<td>Reverses the effect of the previous Undo</td>
</tr>
<tr>
<td>Cut</td>
<td>⌘X Deletes the selected text from the active window and places it on the clipboard.</td>
</tr>
<tr>
<td>Copy</td>
<td>⌘C Copies the selected text to the clipboard.</td>
</tr>
<tr>
<td>Paste</td>
<td>⌘V Pastes the text contents of the clipboard into the active window at the current location.</td>
</tr>
<tr>
<td>Select All</td>
<td>⌘A Highlights all characters in the active window.</td>
</tr>
<tr>
<td>Find…</td>
<td>Enables a search for every occurrence of a specified string (this can include APL glyphs) within the code in the active window (<strong>Session</strong> or <strong>Edit</strong>). For details, see section 4.2.2.</td>
</tr>
<tr>
<td>Find and Replace…</td>
<td>Enables a search for every occurrence of a specified string (this can include APL glyphs) within the code in the active window (<strong>Session</strong> or <strong>Edit</strong>); optionally, a replacement string can be applied on an individual basis. For details, see section 4.2.2.</td>
</tr>
<tr>
<td>Start Dictation…</td>
<td><em>inserted by Apple – not supported by Dyalog Ltd</em></td>
</tr>
<tr>
<td>Emoji &amp; Symbols…</td>
<td><em>inserted by Apple – not supported by Dyalog Ltd</em></td>
</tr>
</tbody>
</table>

### 3.1.4 View Menu

The options available under the **View** menu are detailed in Table 4. These enable the appearance of the Dyalog Session to be changed.

**Table 4. View menu options**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Language Bar</td>
<td>Toggles display of the language bar (see section 3.3.2) at the top of the <strong>Session</strong> window.</td>
</tr>
<tr>
<td>Show Status Bar</td>
<td>Toggles display of the status bar (see section 3.3.4) at the bottom of the <strong>Session</strong> window.</td>
</tr>
<tr>
<td>Show Workspace Explorer</td>
<td>Toggles display of the workspace explorer (see section 3.3.5) to the left of the <strong>Session</strong> window.</td>
</tr>
</tbody>
</table>
Show Debug Toggles display of the debug information window (see section 3.3.6) to the right of the Session window.

Line Wrapping in Session Toggles line wrapping. When off, ⎕PW is used to determine line length. When on, a combination of ⎕PW and line wrapping is used.

Show Status Window Toggles display of the Status window, which displays system messages and supplementary information.

Auto Status When checked, displays the Status window when a new message is generated for it.

Stops Toggles display of an additional column at the left-hand side of the Edit/Trace windows in which break-points can be set/unset. Hiding this column does not remove any previously-set break-points.

Line Numbers Toggles display of line numbers in the Edit/Trace windows.

Outline Toggles code folding/outlining for control structures (including :Section structures) and functions in Edit windows. When toggled, existing code in an open Edit window is automatically updated to reflect the new rules.

Increase Font Size ⌘= Increases the size of the font in all the windows

Decrease Font Size ⌘- Decreases the size of the font in all the windows

Reset Font Size ⌘0 Sets the size of the font in all the windows to its default value.

Toggle Full Screen ^⌘F Toggles the entire session between its current size and full screen size.

### 3.1.5 Window Menu

The options available under the Window menu are detailed in Table 5. This closes all open Edit and Trace windows.

**Table 5. Window menu options**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close All Windows</td>
<td>Closes all open Edit and Trace windows (the Session window, workspace explorer and debug information window remain open).</td>
</tr>
<tr>
<td>&lt;caption&gt;</td>
<td>The Session’s caption (see section 3.3.1).</td>
</tr>
</tbody>
</table>
3.1.6 Action Menu

The options available under the Action menu are detailed in Table 6. These enable Edit and Trace windows to be opened and allow currently-running APL code to be interrupted with trappable events.

**Table 6. Actions menu options**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edit</strong></td>
<td>⇧⏎</td>
</tr>
<tr>
<td></td>
<td>If the cursor is on or immediately after <code>&lt;object name&gt;</code>, then opens an Edit window on that name.</td>
</tr>
<tr>
<td><strong>Trace</strong></td>
<td>⌃⏎</td>
</tr>
<tr>
<td></td>
<td>In the Session window:</td>
</tr>
<tr>
<td></td>
<td>• If the cursor is on a line containing calls to multi-line functions, a Trace window is opened and the functions traced (explicit trace).</td>
</tr>
<tr>
<td></td>
<td>• If the cursor is on a line containing no text and there is a suspended function (or operator) on the execution stack, open a Trace window for that function (naked trace)</td>
</tr>
<tr>
<td></td>
<td>In a Trace window:</td>
</tr>
<tr>
<td></td>
<td>• Open a new Trace window for any multi-line function (or operator) in that line and trace that line as it is evaluated.</td>
</tr>
<tr>
<td>Clear all trace/stop/monitor</td>
<td>Removes any trace/stop/monitor flags (as set by ⎕TRACE/⎕STOP/⎕MONITOR) from all functions in the workspace.</td>
</tr>
<tr>
<td>Weak Interrupt</td>
<td>Suspends execution at the start of the next line.</td>
</tr>
<tr>
<td>Strong Interrupt</td>
<td>Suspends execution after the current primitive operation.</td>
</tr>
</tbody>
</table>

3.1.7 Threads Menu

The options available under the Threads menu are detailed in Table 7. These are only relevant when connected to a compatible interpreter. For more information on threads, see the Dyalog Programming Reference Guide.

**Table 7. Threads menu options**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pause on Error</td>
<td>Pauses all other threads when a thread suspends due to an error or an interrupt.</td>
</tr>
<tr>
<td>Pause all Threads</td>
<td>Pauses all threads (including currently-suspended threads).</td>
</tr>
<tr>
<td>Unpause all threads</td>
<td>Unpauses all paused threads (suspended threads remain suspended).</td>
</tr>
<tr>
<td>Continue all threads</td>
<td>Resumes all paused threads, restarts all suspended threads, and closes the Trace window.</td>
</tr>
</tbody>
</table>
3.1.8 Help Menu

The options available under the Help menu are detailed in Table 8. These provide access to the Dyalog documentation, website, forum and update portal.

At the top of the Help menu is a Search field; this links to the macOS help centre.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting Started</td>
<td>Opens your default web browser on the Getting Started page of Dyalog Ltd’s website.</td>
</tr>
<tr>
<td>Dyalog Help</td>
<td>Opens your default web browser on the Welcome page of Dyalog’s online documentation.</td>
</tr>
<tr>
<td>Language Elements</td>
<td>Opens your default web browser on the Language Elements page of Dyalog’s online documentation.</td>
</tr>
<tr>
<td>Documentation Centre</td>
<td>Opens your default web browser on the Documentation Centre page of Dyalog Ltd’s website.</td>
</tr>
<tr>
<td>Dyalog Website</td>
<td>Opens your default web browser on the home page of Dyalog Ltd’s website.</td>
</tr>
<tr>
<td>Email Dyalog</td>
<td>Opens your email client with a pre-populated template for an email to Dyalog support (including subject line and IDE/interpreter information).</td>
</tr>
<tr>
<td>Latest Enhancements</td>
<td>Opens your default web browser on the Key Features page of Dyalog’s online documentation.</td>
</tr>
<tr>
<td>Read Me</td>
<td>Opens your default web browser on the Dyalog v18.2 ReadMe page, containing the latest information about Dyalog v18.2.</td>
</tr>
<tr>
<td>Third Party Licences</td>
<td>Opens your default web browser on the Licences for Third-Party Components page of Dyalog’s online documentation.</td>
</tr>
</tbody>
</table>

3.2 Keyboard Key Mappings for APL Glyphs

A set of keyboard key mappings for APL glyphs is installed with Dyalog; these are shown in Appendix A. When Dyalog is the active application, these key mappings are automatically enabled. Dyalog attempts to identify your locale and use the appropriate key mappings; if the locale cannot be identified or the locale-specific key mappings has not been configured, then the default configuration is used (key mappings for a US keyboard).

Using the default keyboard, APL glyphs are entered by pressing the prefix key followed by either the appropriate key or the SHIFT key with the appropriate key. The prefix key and key mappings can be customised (see section 7.2.2).

Alternatively, the Dyalog application-specific keyboard mapping can be replaced with a system-wide mapping, which allows Dyalog glyphs to be entered in other applications (for example, email) when it is selected in the main menu bar. For information on installing/enabling a locale-specific set of key mappings, see the Dyalog for macOS Installation and Configuration Guide.
3.3 Session User Interface

By default, the Dyalog Session user interface includes four elements, as shown in Figure 2:

- a caption (see section 3.3.1)
- a language bar (see section 3.3.2)
- a Session window (see section 3.3.3)
- a status bar (see section 3.3.4)

![Figure 2. The Dyalog Session user interface (default)](image)

Additional windows can also be present, as shown in Figure 3:

- a workspace explorer (see section 3.3.5)
- a debug information window (see section 3.3.6)

![Figure 3. The Dyalog Session user interface with workspace explorer and debug information window displayed](image)
3.3.1 Caption

The caption at the top of the Session window is the name of the workspace.

The caption can be customised (see section 7.2.6).

3.3.2 Language Bar

The language bar is located at the top of the Session window, beneath the caption. It contains buttons for each of the glyphs used as primitives in Dyalog.

When the cursor is positioned over one of the glyphs, information for that glyph is displayed. This includes the name of the glyph, the keyboard shortcut to enter it, its monadic/dyadic name and examples of its syntax, arguments and result.

When the mouse is clicked on one of the glyphs, that glyph is copied into the active Session/Edit window at the position of the input cursor (the same as typing it directly into the Session/Edit window).

The order of glyphs in the language bar can be customised by using the mouse to drag-and-drop individual glyphs to the required location.

Display of the language bar can be toggled with the View > Show Language Bar menu option or by entering the Toggle Language Bar command (CtrlL).

On the right hand side of the Language bar is the button:

- Positioning the cursor over the button displays a dynamic tooltip showing all configured keyboard shortcuts for command codes – for more information on these, see section 5.1.
- Clicking on the button displays the Preferences dialog box (the same as selecting the Edit > Preferences menu option) – for more information on the Preferences dialog box, see section 7.2.

3.3.3 Session Window

The primary purpose of the Session window is to provide a scrolling area within which you can enter APL expressions and view results. See section 4.1 for more information on the Session window.

You can move, resize, maximise and minimise the Session window using the standard facilities that macOS provides.

3.3.4 Status bar

The Status bar is located at the bottom of the Session window. It contains a button and seven Session status fields, as shown in figure 4; the items in the Status bar are detailed in table 9.

Figure 4. The Status bar
Table 9. Items in the Status bar

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚙</td>
<td>Opens the Preferences dialog box (see section 7.2).</td>
</tr>
<tr>
<td>&amp;: &lt;number&gt;</td>
<td>Displays the number of threads currently running (minimum value is 1). Turns blue if greater than 1.</td>
</tr>
<tr>
<td>⎕DQ: &lt;number&gt;</td>
<td>Displays the number of events in the APL event queue. Turns blue if non-zero.</td>
</tr>
<tr>
<td>⎕TRAP</td>
<td>Turns blue if ⎕TRAP is set.</td>
</tr>
<tr>
<td>⎕SI: &lt;number&gt;</td>
<td>Displays the length of ⎕SI. Turns blue if non-zero.</td>
</tr>
<tr>
<td>⎕IO: &lt;number&gt;</td>
<td>Displays the value of ⎕IO. Turns blue if not equal to the value defined by the Default_IO configuration parameter (default = 1).</td>
</tr>
<tr>
<td>⎕ML: &lt;number&gt;</td>
<td>Displays the value of ⎕ML. Turns blue if not equal to the value defined by the Default_ML configuration parameter (default = 1).</td>
</tr>
<tr>
<td>Pos &lt;num&gt;/&lt;num&gt;, &lt;num&gt;</td>
<td>Displays the location of the cursor in the active window (line number/total lines, column number).</td>
</tr>
</tbody>
</table>

3.3.5 Workspace Explorer

The workspace explorer is located to the left of the Session window, next to the scrolling area. It contains a hierarchical tree view of all the namespaces, classes, functions, operators and variables in the active workspace (under #) and in the system namespace (under ⎕SE).

Clicking on a namespace, class, function, operator or variable opens its definition in the Edit window.

Display of the workspace explorer can be toggled with the View > Show Workspace Explorer menu option or by entering the Toggle Workspace Explorer command (<WSE>).

3.3.6 Debug Information Window

The debug information window is located to the right of the Session window. It comprises two areas:

- The Threads area lists the threads that are currently in existence. For each thread the following information is provided:
  - a description comprising the thread ID (⎕TID) and name (⎕TNAME)
  - the state of the thread, that is, what it is doing (for example, Session, Pending, :Hold, ⎕NA)
  - the thread requirements (⎕TREQ)
  - a flag indicating whether the thread is Normal or Paused

The toolbar displayed at the top of the Threads area is shown in figure 5; the icons on this toolbar are detailed in table 10.

Figure 5. The Threads area’s toolbar
Table 10. Icons on the Threads area's toolbar

<table>
<thead>
<tr>
<th>Icon</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td>Refresh now</td>
<td>Refresh the list of threads</td>
</tr>
<tr>
<td>➔</td>
<td>Continue execution of this thread</td>
<td>Resume execution of the currently selected thread.</td>
</tr>
<tr>
<td>➔➔</td>
<td>Continue execution of all threads</td>
<td>Resume execution of any paused threads. For information on threads, see the Dyalog Programming Reference Guide.</td>
</tr>
</tbody>
</table>

- The SI Stack area lists the functions in the execution stack; each function in the list also has the line number and source code of the line that caused the function to be added to the stack. Equivalent to the result of )SI.

Display of the debug information window can be toggled with the View > Show Debug menu option.
4 Input Windows

The Dyalog Development Environment can comprise multiple windows:

- **Session** window – created when Dyalog is started and always present while the session is live. There is only one **Session** window.
- **Edit** windows – created and destroyed dynamically as required. There can be multiple **Edit** windows for multiple APL objects.
- **Trace** windows – created and destroyed dynamically as required. There is only one **Trace** window.

When multiple windows are open, the window that has the focus is referred to as the **active** window.

4.1 Session Window

The **Session** window contains:

- the **input line** – the last line entered in the **Session** window; this is (usually) the line into which you type an expression to be evaluated.
- the **Session log** – a history of previously-entered expressions and the results they produced.

If a log file is being used, then the Session log is loaded into memory when Dyalog is started. When Dyalog is closed, the Session log is written to the log file, replacing its previous contents.

4.2 Edit Window

This section applies to the built-in editor. A different editor can be specified by setting the RIDE_EDITOR configuration parameter to the fully-qualified path of the desired editor's executable file.

An **Edit** window is used to define new objects as well as amend existing objects.

An **Edit** window can be opened from the **Session** window in any of the following ways:

- Enter **)ED** `<object name>`
- Enter **[ED]** `'<object name>'`
- Enter `<object name> <ED>`
  
  *(for an explanation of the `<ED>` syntax, see section 5.1)*
- Double-click the cursor over/after `<object name>`

If the object name does not already exist, then it is assumed to be of type function/operator. Different types can be explicitly specified using the **)ED** or **[ED]** options – see **)ED** or **[ED]** in the *Dyalog APL Language Reference Guide* for information.
An Edit window can be opened from the Trace window by entering the Edit command (<ED>), double-clicking the cursor or clicking on the button in the toolbar (see section 4.3.1). The position of the cursor when this is done determines the name of the object that the Edit window if for:
- If the cursor is on or immediately after <object name>, then the Edit window opens on that name.
- If the cursor is anywhere else, then the Edit window opens for the most recently-referenced function in the stack. This is a naked edit.

An Edit window can be opened from another Edit window in any of the following ways:
- Move the cursor over/after <object name> and enter the Edit command (<ED>)
- Double-click with the cursor over/after <object name>

By default, each Edit window is docked to the right of the Session Window.

4.2.1 Toolbar

The toolbar displayed at the top of the Edit window is shown in figure 6; the buttons on this toolbar are detailed in table 11.

![Figure 6. The Edit Window's toolbar](image)

**Table 11. Icons on the Edit window's toolbar**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Cross" /></td>
<td>Save changes and return</td>
<td>Saves changes and closes the Edit Window.</td>
</tr>
<tr>
<td><img src="image" alt="Line Numbers" /></td>
<td>Toggle line numbers</td>
<td>Turns the display of line numbers on/off.</td>
</tr>
<tr>
<td><img src="image" alt="Comment" /></td>
<td>Comment selected text</td>
<td>Add a comment symbol (⍟) to the beginning of the line in which the cursor is positioned. If text has been selected, then a comment symbol is added at the start of the line on which the selection starts and at the start of each subsequent line of text within the selection. Same as the Comment Out command (&lt;AO&gt;).</td>
</tr>
<tr>
<td><img src="image" alt="Uncomment" /></td>
<td>Uncomment selected text</td>
<td>Removes the comment symbol (⍟) at the beginning of the line in which the cursor is positioned. If text has been selected, then comment symbols are removed from the start of each selected line of text. Comment symbols that are not at the start of a line of text cannot be removed. Same as the Uncomment command (&lt;DO&gt;).</td>
</tr>
<tr>
<td><img src="image" alt="Search" /></td>
<td>Search</td>
<td>Opens the Search bar, enabling a search to be performed – see Section 4.2.2.</td>
</tr>
</tbody>
</table>
4.2.2 Search and Replace

The icon in the Edit window’s toolbar opens the Search bar, enabling a search for every occurrence of a specified string (this can include APL glyphs) to be performed within the code in the active Edit window; optionally, a replacement string can be applied on an individual basis.

The Search bar is shown in figure 7; the icons on this toolbar are detailed in table 12.

![Figure 7. The Search bar (default)](image)

Clicking on the icon (Toggle replace mode) extends the Search bar to include a Replace field and related icons as shown in figure 8; the icons on this toolbar are detailed in table 12.

![Figure 8. The Search bar (including display of the Replace fields)](image)

<table>
<thead>
<tr>
<th>Icon</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>←</td>
<td>Search for previous match</td>
<td>Positions the cursor at the previous occurrence of the Search text.</td>
</tr>
<tr>
<td>→</td>
<td>Search for next match</td>
<td>Positions the cursor at the next occurrence of the Search text.</td>
</tr>
<tr>
<td></td>
<td>Find in selection</td>
<td>Only searches within the selected text.</td>
</tr>
<tr>
<td></td>
<td>Close</td>
<td>Closes the Search bar.</td>
</tr>
<tr>
<td></td>
<td>Replace</td>
<td>Replaces the selected text in the Edit window with the text specified in the Replace field of the Search bar and highlights the next match.</td>
</tr>
<tr>
<td></td>
<td>Replace all</td>
<td>Replaces all occurrences of the text specified in the Find field of the Search bar with that specified in the Replace field of the Search bar.</td>
</tr>
</tbody>
</table>

The Find field includes three filters, as detailed in table 13. Any combination of these filters can be applied when performing a search.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aa</td>
<td>Match case</td>
<td>Applies a case-sensitive filter so that only occurrences that match the case of the string in the Find field are highlighted.</td>
</tr>
<tr>
<td></td>
<td>Match whole word</td>
<td>Only highlights whole words that match the string in the Find field. <strong>Note:</strong> Some glyphs are treated as punctuation when identifying “whole words”, for example, ; : , { } [ ] ( and ).</td>
</tr>
<tr>
<td>J*</td>
<td>Use regular expression</td>
<td>Allows regular expressions to be specified in the Find field.</td>
</tr>
</tbody>
</table>
To search for a string:

1. Enter the string to search for in the Find field in one of the following ways:
   - Press the Search button and enter the string directly in the Find field.
   - Enter the Search command (<SC>) and enter the string directly in the Find field.
   - Select the string in the Edit window or position the cursor over a word and enter the Search command (<SC>) or press the Search button; the selected string or the word under the cursor is copied to the Find field.

   All occurrences of the specified string are highlighted in the Edit window. If the content of the Edit window is sufficiently long for there to be a vertical scroll bar, then the locations of occurrences of the specified string within the entire content are identified by yellow marks overlaid on the scroll bar.

2. Press the Enter key to select the first occurrence of the search string after the last position of the cursor.

3. Press the Search for next match button to advance the selection to the next occurrence of the search string.

4. Repeat step 1 as required (the search is cyclic).

5. Use the Esc key to exit the search functionality.

To replace a string:

1. Do one of the following:
   - Enter the string to be replaced in the Find field in one of the following ways:
     - Press the Search button and enter the string directly in the Find field.
     - Enter the Search command (<SC>) and enter the string directly in the Find field.
     - Select the string in the Edit window and enter the Search command (<SC>) or press the Search button; the selected string is copied to the Find field.

     Press ▶ to extend the Search bar to display the Replace field.
   - Enter the Replace command (<RP>) and enter the string directly in the Find field.
   - Select the string in the Edit window and enter the Replace command (<RP>); the selected string is copied to the Find field and the Replace field is displayed.

2. Enter the replacement string directly in the Replace field.

3. Press the Enter key to select the first occurrence of the search string after the last position of the cursor.
4. Do one of the following:

- Press the **Enter** key or the **Replace** button \( \text{☑} \) to replace the selected occurrence of the search string and to advance the selection to the next occurrence of the search string.
- Press the **Search for next match** button \( \text{→} \) to leave the selected occurrence of the search string unaltered and to advance the selection to the next occurrence of the search string.
- Press the **Replace all** button \( \text{☑} \) to replaces all occurrences of the search string.

5. Press the **Close** button \( \text{x} \) to exit the search and replace functionality.

### 4.2.3 Exiting the Edit Window

To save changes and close the **Edit** window:

- click on \( \text{✗} \) in the **Edit** window’s tab and select **yes** when prompted whether to save changes
- click on \( \text{☐} \) in the **Edit** window’s toolbar
- use the **Escape** command (\(<\text{EP}>\))

To close the **Edit** window without saving changes:

- click on \( \text{✗} \) in the **Edit** window’s tab and select **no** when prompted whether to save changes
- use the **Quit** command (\(<\text{QT}>\))

### 4.3 Trace Window

A **Trace** Window aids debugging by enabling you to step through your code line by line, display variables in **Edit** windows and watch them change as the execution progresses. Alternatively, you can use the **Session** window and **Edit** windows to experiment with and correct your code.

A **Trace** window can be opened from the **Session** window by entering \(<\text{expression}>\text{<TC>}\). This is an **explicit trace** and lets you step through the execution of any non-primitive functions/operators in the expression.

If there is a suspended function in the stack, then a **Trace** window can be opened from the **Session** window by double-clicking on a blank line – this can be a blank input line or a blank line in the Session log. A **Trace** window for the most recently-referenced function in the stack is displayed; this is a **naked trace**.

By default, Dyalog is also configured to initiate an **automatic trace** whenever an error occurs, that is, the **Trace** window opens and becomes the active window and the line that caused the execution to suspend is selected. This is controlled by the **TRACE_ON_ERROR** configuration parameter (for information on configuration parameters, see the **Dyalog for macOS Installation and Configuration Guide**).

By default, each **Trace** window is docked beneath the **Session** and **Edit** windows. Other than setting/removing breakpoints (see section 5.9.1), **Trace** windows are read-only.
4.3.1 Toolbar

The toolbar displayed at the top of the Trace window is shown in figure 9; the buttons on this toolbar are detailed in table 14.

![Figure 9. The Trace Window's toolbar](image)

<table>
<thead>
<tr>
<th>Icon</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Quit this function</td>
<td>Cuts the execution stack back one level.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Toggle line numbers</td>
<td>Turns the display of line numbers on/off.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Execute line</td>
<td>Executes the current line and advances to the next line.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Trace into expression</td>
<td>Traces execution of the current line and advances to the next line. If the current line calls a user-defined function then this is also traced.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Stop on next line of calling function</td>
<td>Continues execution of the code in the Trace window from the current line to completion of the current function or operator. If successful, the selection advances to the next line of the calling function (if there is one).</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Continue execution of this thread</td>
<td>Closes the Trace window and resumes execution of the current application thread from the current line.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Continue execution of all threads</td>
<td>Closes the Trace window and resumes execution of all suspended threads.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Interrupt</td>
<td>Interrupts execution with a weak interrupt.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Edit name</td>
<td>Converts the Trace window into an Edit window as long as the cursor is on a blank line or in an empty space. However, if the cursor is on or immediately after an object name that is not the name of the suspended function, then an Edit window for that object name is opened.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Clear stops for this object</td>
<td>Clears all breakpoints (resets ⎕STOP) on the function(s) in the Edit/Trace windows.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Search</td>
<td>Opens the Search bar, enabling a search to be performed – see Section 4.3.2.</td>
</tr>
</tbody>
</table>

4.3.2 Search

The icon in the Trace window’s toolbar opens the Search bar, enabling a search for every occurrence of a specified string (this can include APL glyphs) to be performed within the code in the Trace window.

The Search bar is shown in figure 10; the icons in the Search bar are detailed in table 15.
Figure 10. The Search bar

Table 15. Icons on the Search bar

<table>
<thead>
<tr>
<th>Icon</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>←</td>
<td>Search for previous match</td>
<td>Positions the cursor at the previous occurrence of the Search text.</td>
</tr>
<tr>
<td>→</td>
<td>Search for next match</td>
<td>Positions the cursor at the next occurrence of the Search text.</td>
</tr>
<tr>
<td></td>
<td>Find in selection</td>
<td>Only searches within the selected text.</td>
</tr>
<tr>
<td></td>
<td>Close</td>
<td>Closes the Search bar.</td>
</tr>
</tbody>
</table>

To search for a string:

1. Enter the string to search for in the Find field in one of the following ways:
   - Press the Search button Q and enter the string directly in the Find field.
   - Enter the Search command (<SC>) and enter the string directly in the Find field.
   - select the string in the Trace window and enter the Search command (<SC>) or press the Search button Q; the selected string is copied to the Find field.

   All occurrences of the specified string are highlighted in the Trace window and the number of occurrences is displayed to the right of the Find field. If the content of the Trace window is sufficiently long for there to be a vertical scroll bar, then the locations of occurrences of the specified string within the entire content are identified by yellow marks overlaid on the scroll bar.

2. Press the Enter key to select the first occurrence of the search string after the last position of the cursor.

3. Press the Search for next match button → to advance the selection to the next occurrence of the search string.

4. Repeat step 1 as required (the search is cyclic).

5. Press the Close button X to exit the search functionality.

4.3.3 Exiting the Trace Window

To close the Trace window:
   - click on X in the Trace window’s tab
   - click on ⌘ in the Trace window’s toolbar
   - use the Escape command (<EP>)
   - use the Quit command (<QT>)
   - press the Esc key

When the Trace window is closed, the function within that Trace window is removed from the stack. It is possible to close all Trace/Edit windows without clearing the stack by selecting the Window > Close All Windows menu option or using 2023 XYZ – see the Dyalog APL Language Reference Guide.
5 Working in a Dyalog Session

The main purpose of a development environment is to enable you to enter and execute expressions; this chapter describes how this can be achieved and explains the functionality that is provided to simplify the process.

5.1 Keyboard Shortcuts and Command Codes

Keyboard shortcuts are keystrokes that generate an action rather than produce a symbol. Dyalog supports numerous shortcut keys, each of which are identified by a command code; for example, the action to open the Trace window is identified by the code TC (described in the documentation as <TC>) and mapped to keystrokes. For a complete list of the command codes that can be used in a Dyalog Session and the keyboard shortcuts for those command codes, see Appendix B.

Positioning the cursor over the button on the right hand side of the Language bar displays a dynamic tooltip showing all configured keyboard shortcuts for command codes. Clicking on the button displays the Preferences dialog box (the same as selecting the Edit > Preferences menu option), through which keyboard shortcuts can be customised (see section 7.2.3).

Additional macOS-specific keyboard shortcuts are documented at https://code.visualstudio.com/shortcuts/keyboard-shortcuts-macos.pdf – these are all valid within a Dyalog Session with the exception of the following, which have been removed due to clashes with the Dyalog keyboard layout:
  • ⌘⌥I – Insert cursor at end of each line selected
  • ⌘⌥A – Toggle block comment
  • ⌘⌥F – Format document

5.2 Accented Characters

macOS includes functionality whereby pressing and holding a key displays accented or alternative versions of the letter on that key, any of which can then be selected. However, only certain accented characters are supported in names – for the full list, see the Dyalog Programming Reference Guide. For example, holding down the a key presents options of à, á, â, ã, ã, ã and ā, but of these eight options, only the first seven are supported in names.

5.3 Navigating the Windows

When multiple windows are open, the window that has the focus is referred to as the active window. A window can be made the active window by clicking within it.

The Session, Edit and Trace windows form a closed loop for the purpose of navigation:
  • to make the next window in this loop the active window, enter the Tab Window command (<TB>)
• to make the previous window in this loop the active window, enter the Back Tab Window command (<BT>)

An active Edit/Trace window can be closed after changes have been made to its content:
• to save any changes in the content of the active window before closing it, enter the Escape command (<EP>), press the Close button in its menu bar or, if the window is docked, press in the window’s tab
• to discard any changes in the content of the active window before closing it, enter the Quit command (<QT>)

5.4 Display of Windows

By default, the Trace window and Edit windows are docked beneath and to the right of the Session window respectively. If the menu option View > Show Workspace Explorer is checked, then the workspace explorer is docked to the far left of any open windows; if the menu option View > Show Debug is checked, then the debug information window is docked to the far right of any open windows.

Docked windows can be selected (by clicking within them), resized (by moving the splitter bar) and maximised/minimised (using the “traffic light” buttons).

New Edit and Trace windows can be floating rather than docked by selecting the Floating windows checkbox in the Windows tab of the Preferences dialog box (see section 7.2.8).

Clicking in the tab of any window enables that window to be dragged to a different location.

5.5 Entering APL Characters

APL glyphs can be entered in a Dyalog Session by:
• typing the glyph in the Session window or Edit window using the appropriate key combination (see section 3.2).
• clicking on the appropriate glyph on the language bar (see section 3.3.2) – this copies that glyph into the active Session/Edit window at the position of the input cursor.

When typing a glyph directly rather than using the language bar, if you pause after entering the prefix key then the autocomplete functionality (see section 5.6.2) displays a list of all the glyphs that can be produced. If you enter the prefix key a second time then a list of all the glyphs that can be produced is again displayed but this time with the names (formal and informal) that are used for each glyph.

For example:

```
A default prefix key
```

The autocomplete functionality list includes the following for the ⍟ glyph:

- ⍟ `* `logarithm
- ⍟ `* `naturallogarithm
- ⍟ `* `circlestar
- ⍟ `* `starcircle
- ⍟ `* `splat
This means that you can enter the ⋅ glyph by selecting (or directly typing) any of the following:

`*`  `logarithm`
`naturallogarithm`
`circlestar`
`starcircle`
`splat`

As you enter a name, the autocomplete functionality restricts the list of options to those that match the entered name.

For example, entering:

`ci`

restricts the list to:

`circlestar`
`circular`
`circletile`
`circlebar`
`circlebackslash`
`circlediaeresis`

### 5.6 Entering Expressions

Dyalog provides several mechanisms that assist with accuracy and provide clarity when entering expressions.

#### 5.6.1 Paired Enclosures

*Applicable in the Session window and the Edit window.*

Enclosures in Dyalog include:

- parentheses `( )`
- braces `{ }`
- brackets `[ ]`

Angle brackets `< >` are not enclosures.

When an opening enclosure character is entered, Dyalog automatically includes its closing pair. This reduces the risk of an invalid expression being entered due to unbalanced enclosures. This feature can be disabled in the **General** tab of the Preferences dialog box (see section 7.2.1).

#### 5.6.2 Autocomplete

*Applicable in the Session window and the Edit window.*

Dyalog includes autocomplete functionality for names to reduce the likelihood of errors when including them in an expression (and to save you from having to enter complete names or remember cases for case-sensitive names).

As a name is entered, Dyalog displays a pop-up window of suggestions based on the characters already entered and the context in which the name is being used.
For example, if you enter a character, the pop-up list of suggestions includes all the system functions and variables. Entering further characters filters the list so that only those system functions and variables that start with the exact string entered are included.

When you start to enter a name in the Session window, the pop-up list of suggestions includes all the namespaces, variables, functions and operators that are defined in the current namespace. When you start to enter a name in the Edit window, the pop-up list of suggestions includes all names that are localised in the function header.

To select a name from the pop-up list of suggestions, do one of the following:
- click the mouse on the name in the pop-up list
- use the right cursor key to select the top name in the pop-up list
- use the up and down cursor keys to navigate through the suggestions and the right cursor key or the ENTER key to enter the currently-highlighted name

The selected name is then completed in the appropriate window.

This feature can be disabled or customised in the General tab of the Preferences dialog box (see section 7.2.1).

### 5.6.3 Context-Sensitive Help

*Applicable in the Session window, Edit window and Trace window*

With the cursor on or immediately after any system command, system name, control structure keyword or primitive glyph, enter the Help command (<HLP>). The documentation for that system command, system name, control structure keyword or primitive glyph will be displayed.

### 5.6.4 Syntax Colouring

*Applicable in the Edit window and Trace window*

Syntax colouring assigns different colours to various components, making them easily identifiable. The syntax colouring convention used is detailed in table 16.

**Table 16. Syntax colouring convention**

<table>
<thead>
<tr>
<th>Colour</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>black</td>
<td>global names</td>
</tr>
<tr>
<td>grey</td>
<td>names</td>
</tr>
<tr>
<td></td>
<td>namespaces</td>
</tr>
<tr>
<td></td>
<td>numbers</td>
</tr>
<tr>
<td></td>
<td>tradfn syntax (specifically, header line and final ∇)</td>
</tr>
<tr>
<td>maroon</td>
<td>control structure keywords</td>
</tr>
<tr>
<td>red</td>
<td>errors (including unmatched parentheses, quotes and braces)</td>
</tr>
<tr>
<td>teal</td>
<td>strings</td>
</tr>
<tr>
<td></td>
<td>comments</td>
</tr>
<tr>
<td>navy</td>
<td>primitive functions</td>
</tr>
<tr>
<td></td>
<td>zilde</td>
</tr>
</tbody>
</table>
Syntax colouring can be customised (see section 7.2.5).

5.7 Executing Expressions

5.7.1 Executing a New Expression

Applicable in the Session window

After entering a new expression in the input line, that expression is executed by pressing the Enter key or with the Enter command (<ER>). Following execution, the expression (and any displayed results) become part of the Session log.

5.7.2 Re-executing a Previous Expression

Applicable in the Session window

Instead of entering a new expression in the input line, you can move back through the Session log and re-execute a previously-entered expression.

To re-execute a previously-entered expression:

1. Locate the expression to re-execute in one of the following ways:
   • Scroll back through the Session log.
   • Use the Backward command (<BK>) and the Forward command (<FD>) to cycle backwards and forwards through the input history, successively copying previously-entered expressions into the input line.

2. Position the cursor anywhere within the expression that you want to re-execute and press the Enter key or use the Enter command (<ER>).

If required, a previously-entered expression can be amended prior to execution. In this situation, when the amended expression is executed it is copied to the input line; the original expression in the Session log is not changed. If you start to edit a previous expression and then decide not to, use the Quit command (<QT>) to return the previous expression to its unaltered state.

5.7.3 Re-executing Multiple Previous Expressions

Applicable in the Session window

Multiple expressions can be re-executed together irrespective of whether they were originally executed sequentially (certain system commands cause re-execution to stop once they have been completed, for example, )LOAD and )CLEAR ).

To re-execute multiple previously-entered expressions:

1. Locate the first expression to re-execute in one of the following ways by scrolling back through the Session log.

2. Change the expression in some way. The change does not have to impact the purpose of the expression; it could be an additional space character.
3. Scroll through the Session log to locate the next expression to re-execute and change it in some way. Repeat until all the required expressions have been changed.
4. Press the Enter key or enter the Enter command (<ER>)

The amended expressions are copied to the input line and executed in the order in which they appear in the Session log; the original expressions in the Session log are not changed.

**To re-execute contiguous previously-entered expressions:**
1. Position the cursor at the start of the first expression to re-execute.
2. Press and hold the mouse button (left-click)+ and drag the cursor to the end of the last expression to re-execute.
3. Copy the selected lines to the clipboard using the Copy command (<CP>) or the Cut command (<CT>) or the Copy/Cut options in the Edit menu.
4. Position the cursor in the input line and paste the content of the clipboard back into the Session using the Paste command (<PT>), the Paste option in the Edit menu or the Paste option in the right-click drop-down menu.
5. Press the Enter key or enter the Enter command (<ER>).

This technique can also be used to move lines from the Edit window into the Session window and execute them.

### 5.8 Threads

Dyalog supports multithreading. For information on threads, see the *Dyalog Programming Reference Guide*.

The number of threads currently in use is displayed in the Status bar (see section 3.3.4).

### 5.9 Suspending Execution

To assist with investigations into the behaviour of a set of statements (debugging), the system can be instructed to suspend execution just before a particular statement is executed. This is done using a breakpoint – see section 5.9.1.

It is sometimes necessary to suspend the execution of a function, for example, if an endless loop has been inadvertently created or a response is taking an unacceptably long time. This is done using an interrupt – see section 5.9.2.

Suspended functions are placed in the stack, with the most recently-referenced function at the top of the stack. The content of the stack can be queried with the )SI system command; this generates a list of all suspended and pendent (that is, awaiting the return of a called function) functions, where suspended functions are indicated by a *. For more information on the stack and the state indicator, see the *Dyalog Programming Reference Guide*.

#### 5.9.1 Breakpoints

*Applicable in the Edit window and the Trace window.*

When a function that includes a breakpoint is run, its execution is suspended immediately before executing the line on which the breakpoint is set and the Trace window is automatically opened (assuming that automatic trace is enabled – see section 4.3).
Breakpoints are defined by dyadic `⎕STOP` and can be toggled on and off in an Edit or Trace window by left-clicking on the far left of the line before which the breakpoint is to be applied or by placing the cursor anywhere in the line before which the breakpoint is to be applied and entering the Toggle Breakpoint command `<BP>`.

Note that:
- Breakpoints set or cleared in an Edit window are not established until the function is fixed.
- Breakpoints set or cleared in a Trace window are established immediately.

When a breakpoint is reached during code execution, event 1001 is generated; this can be trapped. For more information, see `⎕TRAP` in the Dyalog APL Language Reference Guide.

5.9.2 Interrupts

Dyalog responds to both strong and weak interrupts.

Entering a strong interrupt suspends execution as soon as possible (generally after completing execution of the primitive currently being processed). A strong interrupt is issued by selecting Actions > Strong Interrupt in the menu options or by entering the Strong Interrupt command `<SI>`.

Entering a weak interrupt suspends execution at the start of the next line (generally after completing execution of the statement currently being processed). A weak interrupt is issued by selecting Actions > Weak Interrupt in the menu options or by entering the Weak Interrupt command `<WI>`.

When a strong or weak interrupt is issued during code execution, event 1003 or 1002 (respectively) is generated; these can be trapped. For more information, see `⎕TRAP` in the Dyalog APL Language Reference Guide.

5.10 Terminating a Session

The Dyalog Session can be terminated (without having to close any open windows first) in any of the following ways:
- Select Dyalog > Quit Dyalog in the menu options
- Enter ⌘Q

In addition, when the Session window is the live window, the Dyalog Session can be terminated cleanly in any of the following ways:
- Select Window > Close window in the menu options
- Enter ⌘W
- Enter <QIT>
- Enter )off
- Click on the red "traffic light" close button
6 Unsupported Language Elements

When running Dyalog for macOS, a few of the features that are available with other operating systems do not function as might be expected.

6.1 Function Key Configuration

Character strings (including command keys) can be associated with programmable function keys using the ⎕PFKEY system function. When running a Dyalog Session through on macOS, ⎕PFKEY can be used to define/display the keystrokes for a designated function key; however, that function key does not acquire the defined set of keystrokes, rendering ⎕PFKEY of no real use. Instead, function keys should be set through the Shortcuts tab of the Preferences dialog box (see Section 7.2.3).

6.2 Underscored Characters in Window Captions

Underscored characters can be entered into the Session window and Edit windows using the `\_<letter>` method, for example, enter `\_f` to produce Ⓐ.

However, an operating system restriction means that, in window captions:
- if the APL385 font is installed (and the operating system has been configured to allow the title bar to use it), underscored characters are displayed as circled characters
- if the APL385 font is not installed, underscored characters are displayed as a white rectangle, a black rectangle containing a question mark, or some other Unicode-compliant substitution.

For example:
Open an Edit window for an object that has an underscored name:

)ed A

The Edit window that is opened displays the name correctly, but its title (caption) is displayed incorrectly, as shown (assuming window is docked):

![Image]

6.3 Operating System Terminal Window Interaction

Features that rely on interaction with an operating system terminal window (that is, ⎕SR and )SH or )CMD with no argument) do not work in a Dyalog Session running under Dyalog for macOS. Instead:
- ⎕SR generates a trappable error
- )SH or )CMD with no argument produces a "Feature disabled in this environment" message.
7 Customising Your Session

The appearance and behaviour of a Dyalog Session can be customised to meet personal preferences and corporate guidelines. Configuration can be performed:

- through the View menu – see section 7.1
- through the Preferences dialog box – see section 7.2
- using configuration parameters – see section 7.3

Customisations performed using any of these methods persist between Sessions (they also persist when the installed version of Dyalog is upgraded).

To remove all customisations and return to the initial default settings, rename/delete the `$HOME/Library/Application Support/ride<version>` directory (hidden directory – access from the command line). This resets all RIDE-specific settings (does not reset configuration parameter).

7.1 View Menu

The View menu (see section 3.1.4) includes options that enable the appearance of the Dyalog Session to be changed. Select these options to:

- show/hide the language bar (see section 3.3.2)
- show/hide the status bar (see section 3.3.4)
- show/hide the workspace explorer (see section 3.3.5)
- show/hide the debug information window (see section 3.3.6)
- change the font size in all windows in the Session
- configure the display/functionality of the Edit/Trace windows

7.2 Preferences Dialog Box

The Preferences dialog box can be used to customise:

- the automatic formatting of code in an Edit window (see section 7.2.1)
- the default keyboard key mappings for APL glyphs (see section 7.2.2)
- the keyboard shortcuts for command codes (see section 7.2.3)
- the saved responses for dialog box options (see section 7.2.4)
- the syntax colouring and background colour (see section 7.2.5)
- the caption of the Session window (see section 7.2.6)
- the menu options presented by the menu bar (see section 7.2.7)
- the display of floating Edit/Trace windows (see section 7.2.8)

The Preferences dialog box can be opened in either of the following ways:

- selecting the Dyalog > Preferences menu option
- clicking on the Preferences button on the right hand side of the Language bar
- entering the Show Preferences command (`<PRF>`)

The Preferences dialog box comprises multiple tabs. A button enables the contents of the Keyboard and Shortcuts tabs to be printed.
7.2.1 General Tab

Allows customisation of a variety of features.

To change the general configuration options

1. Open the General tab.
2. Select the appropriate check boxes and set the variables as required:
   
   - **Indent content when an editor is opened**
     Select this to apply the indentation rules to the contents of an Edit window when it is opened.
   
   - **Handle formatting through the interpreter**
     Select this to enforce default code formatting, ensuring consistency with previous version of Dyalog and avoiding formatting changes that could be detected by source code management systems when multiple machines are used to maintain a body of code.
   
   - **Auto-indent <number> spaces**
     Only relevant when Handle formatting through the interpreter is not selected. Select this and define the number of spaces by which each level of nested code should be indented relative to the previous level of indentation in the Edit window. If not selected, code in the Edit window will be left justified.
     - in methods: <number> spaces
       Only relevant when Handle formatting through the interpreter is not selected. Select this and define the number of spaces by which the contents of tradfns should be indented relative to the start/end ∇ glyphs.

   When changes to auto-indentation are applied, the indentation of existing code in an open Edit window does not change unless you enter the Reformat command (<RD>) when the Edit window is the active window. However, any new code entered in the Edit window follows the new rules.

   - **Indent lines that contain only a comment**
     Only relevant when Handle formatting through the interpreter is not selected. Select this to apply the appropriate indentation to lines that start with the ⍝ glyph. If this is not selected, then lines that start with the ⍝ glyph remain as positioned.

   - **Highlight matching brackets: () [] {}**
     Select this to automatically highlight the matching start and end enclosures when positioning the cursor before or after one of them (with contiguous brackets, the bracket immediately before the cursor has its other enclosure highlighted).

   - **Auto-close brackets**
     Select this to automatically add the paired enclosure when an opening enclosure is entered (see section 5.6.1).

   - **Autocompletion**
     Select an autocompletion option from the drop-down list (autocompletion makes suggestions when entering the name of, for example, namespaces, variables, functions, operators, user commands and system names). Options include:
     - off – disables autocomplete functionality.
• classic – displays a pop-up window of suggestions based on the characters already entered and the context in which the name is being used (see section 5.6.2).
• shell – resembles the autocomplete functionality of the Linux bash shell in its use of the tab key.

• **Autocompletion after `<time>` ms**
  Specify a time interval after which the RIDE’s autocompletion functionality is activated.

• **Autocomplete control structure with snippets**
  Select this to be presented with auto-completion template options for control structures in the **Edit** window.

• **Highlight matching words**
  Select this to highlight all occurrences of a selected string in the same window.

• **Show value tips**
  Select this to display the referent of a name. When the cursor is positioned over or immediately after a name, the name is highlighted and its referent is displayed (for example, the value of a variable or the body of a function).

• **Show tips for glyphs**
  Select this to show the tooltip for a glyph. When the cursor is positioned over or immediately after a glyph, the glyph is highlighted and information about it is displayed – this includes the name of the glyph, the keyboard shortcut to enter it, its monadic/dyadic name and examples of its syntax, arguments and result.

• **Auto PW**
  Select this to display text in the **Session** window up to the full width of the **Session** window before wrapping (automatically updates when the Session is resized). Also sets `⎕PW` to this value.

• **Persistent history `<number>` lines**
  Select this and define the number of lines that are available to recall using the `Backward/Undo` command (`<BK>`). This specifies how many input lines RIDE remembers from the end of one RIDE session to the start of the next session.

• **Block cursor**
  Select this to display the cursor as a solid rectangular block rather than a vertical line.

• **Cursor blinking**
  Select a cursor animation from the drop-down list (for example, blink or solid).

• **Highlight current line**
  Select an option from the drop-down list to indicate how the line that the cursor is currently positioned on should be emphasised (applies to all windows). Options include:
  • none – do not indicate the current line.
  • gutter – display a small grey rectangle in the first column of the line.
- line – display a grey rectangle around the entire line except the first column.
- all – gutter and line.

- Minimap enabled
  Select this to display a dynamic impression of the entire (or very large portion of) contents of the Session/Edit window in a column on the right-hand side of that window. Clicking within the minimap moves the display to that location.

- Minimap render characters
  Only relevant when Minimap enabled is selected. Select this to use tiny rendered font in the minimap rather than greeking (showing a representation of the text).

- Minimap show slider
  Only relevant when Minimap enabled is selected. Select an option from the drop-down list to indicate how the currently-displayed content of the window is highlighted in the minimap. Options include:
  - always – the current display is always highlighted in the minimap.
  -mouseover – the current display is only highlighted in the minimap when the mouse is positioned over the minimap.

- Double click to edit
  Select this to open the Edit window when double-clicking within a word rather than selecting/highlighting the whole word.

- Show quit prompt
  Select this to display a confirmation dialog box when exiting the RIDE session.

- Show toolbar in editor/trace windows
  Select this to display the toolbar in the Edit/Trace windows.

- Connect on quit
  Select this to be returned to the RIDE-Dyalog Session dialog box on exiting a Session.

- Include filename in editor title
  Select this to include an object's associated filename in the caption at the top of the Edit window.

3. Click on OK to save your changes and close the Preferences dialog box, Apply to save your changes without closing the Preferences dialog box or Cancel to close the Preferences dialog box without saving your changes.

Settings that impact the automatic reformatting of code can cause changes to whitespace – this can be interpreted as changes to the source code. This means that:

- opening a scripted object in the Edit window can cause the source of that object to change (when closing an Edit window, you might be prompted to save a function even though you have not made any changes to it).
- viewing an object can change its file timestamp; source code management systems can subsequently report changes due to the changed file timestamp.
source code changes resulting from reformatting will be evident in the results of system functions such as ⎕AT, ⎕SRC, ⎕CR, ⎕VR and ⎕NR.

7.2.2 Keyboard Tab

Allows customisation of the default keyboard key mappings for APL glyphs (see section 3.2). This is only relevant if a locale-specific keyboard has not been installed.

To replace the keyboard with a locale-specific keyboard in the Session, or to enter Dyalog glyphs in other applications (for example, email), see the Dyalog for macOS Installation and Configuration Guide.

An example of the default keyboard that is installed with Dyalog for use in a Session is shown in Appendix A (this is the UK keyboard – other locales are available through a drop-down list).

To customise the default keyboard’s Prefix key

1. Open the Keyboard tab and select the appropriate keyboard from the drop-down list of options.
2. In the Prefix key field, enter the new prefix key (by default this is `).

   In locales in which ` is a dead key, $ is a viable alternative.

Be careful when selecting a new prefix key – although there are no restrictions, choosing certain keys (for example, alphanumeric characters) would restrict the information that could be entered in a Session.

3. Click on OK to save your changes and close the Preferences dialog box, Apply to save your changes without closing the Preferences dialog box or Cancel to close the Preferences dialog box without saving your changes.

To customise the default keyboard’s key mappings

1. Open the Keyboard tab and select the appropriate keyboard from the drop-down list of options.
2. In the image of a keyboard, select the glyph to be replaced.
3. In the language bar, click on the glyph to replace the selected glyph with.
4. Repeat steps 2 and 3 until the key mappings are as required.
5. Click on OK to save your changes and close the Preferences dialog box, Apply to save your changes without closing the Preferences dialog box or Cancel to close the Preferences dialog box without saving your changes.

Your selection of keyboard key mappings is unrestricted – you can choose to map the same glyph to multiple keys or have glyphs that are not represented on the keyboard at all. For example, when dealing with dfns on a Danish keyboard, it might be convenient to map { and } to a simpler key combination.
7.2.3 Shortcuts Tab

Allows customisation of the keyboard shortcuts for command codes (see section 5.1 and Appendix B). Multiple shortcuts can be defined for any command code, but each shortcut must be unique.

To change the keyboard shortcut for a command code

1. Open the Shortcuts tab.
2. Locate the command code that you want to define a new keyboard shortcut for. This can be done by scrolling through the list of possible command codes or by entering a string in the Search field. If a string is entered in the Search field then a dynamic search of both the command codes and descriptions is performed.
3. Optionally, delete any existing shortcuts for the command by clicking on the red cross to the right of the shortcut.
4. Optionally, add a new shortcut. To do this:
   a. Click on the plus symbol that appears to the right of any existing shortcuts when the shortcut is moused over. A field in which to enter the shortcut is displayed.
   b. In this field, enter the keystrokes to map to this action. The field closes when the keystrokes have been entered and the new shortcut is displayed on the Shortcuts tab.
      If the keystrokes that you enter are already used for a different command code, then both occurrences will be highlighted and you should remove any duplicate entries (an error message will be displayed if you attempt to apply/save settings that contain duplicate entries).
5. Click on OK to save your changes and close the Preferences dialog box, Apply to save your changes without closing the Preferences dialog box or Cancel to close the Preferences dialog box without saving your changes.

The tooltip showing keyboard shortcuts for command codes (obtained by positioning the cursor over the button on the right hand side of the Language bar) is dynamically updated to any customisation.

7.2.4 Saved Responses Tab

Allows saved option selections to be deleted.

When using Link, options selected in dialog boxes are recorded in this tab. For example, saving file content opens a dialog box that offers options of fixing as code in the workspace, saving the file to disk, or discarding changes; the selected option for this file (or, optionally, all files with the same file extension) is recorded in this tab.

To delete a previously-saved option selection for a file

1. Open the Saved Responses tab.
2. Select the file for which you want to delete the saved option.
3. Click Delete.
4. Click on OK to save your changes and close the Preferences dialog box, Apply to save your changes without closing the Preferences dialog box or Cancel to close the Preferences dialog box without saving your changes.
7.2.5 Colours Tab

Allows customisation of the syntax colouring (see section 5.6.4). Several schemes are provided, any of which can be used as they are or further customised.

To change the syntax colouring to a predefined scheme

1. Open the Colours tab.
2. In the Scheme field, select the syntax colouring scheme that you want to use. When a scheme is selected, the example shown is updated to use that colour scheme.
3. Click on OK to save your changes and close the Preferences dialog box, Apply to save your changes without closing the Preferences dialog box or Cancel to close the Preferences dialog box without saving your changes.

To define a new syntax colouring scheme

1. Open the Colours tab.
2. In the Scheme field, select the syntax colouring scheme that is closest to the scheme that you want to define. When a scheme is selected, the example shown is updated to use that colour scheme.
3. Click on Clone.
   A copy is made of the selected scheme and additional fields are displayed to allow customisation of the scheme's name and syntax colouring.
4. Define your colour scheme. To do this:
   a. Select the element that you would like to change the display of from the drop-down list or by clicking in the example. The customisation fields reflect the current settings for the selection made.
   b. Select a foreground colour and background (highlighting) colour for that syntax as required. If a background colour is selected, then the slide bar can be used to set its transparency.
   c. Select the appropriate checkboxes to make that syntax bold, italic or underlined as required.
   d. Repeat as required.
5. Optionally, rename your colour scheme. To do this:
   a. Click on Rename.
      The name in the Scheme field becomes editable.
   b. Edit the name in the Scheme field.
6. Click on OK to save your changes and close the Preferences dialog box, Apply to save your changes without closing the Preferences dialog box or Cancel to close the Preferences dialog box without saving your changes.
7.2.6 Title Tab

Allows customisation of the caption at the top of the Session window (see section 3.3.1).

To change the Session window caption

1. Open the Title tab.

2. In the Window title field, enter the new name for the Session. Some variable options that can be included are listed beneath this field – clicking on these inserts them into the Window title field.

3. Click on OK to save your changes and close the Preferences dialog box, Apply to save your changes without closing the Preferences dialog box or Cancel to close the Preferences dialog box without saving your changes.

The Preferences dialog box is closed and the caption in the Session window changes to reflect the new caption.

The default value is \{WSID\}.

Setting this to \{WSID\} – \{HOST\} : \{PORT\}  (PID: \{PID\}) gives a different result on every machine. For example:
/Users/nick/myws.dws – 127.0.0.1:49376 (PID: 293)

Changing this to \{CHARS\} \{BITS\} gives information that could be the same on multiple machines. For example:
Unicode 64

7.2.7 Menu Tab

Allows customisation of the menu options in the menu bar (see section 3.1).

Great care should be taken when customising the menu options; although the ability to makes changes is provided, it is not an activity that Dyalog Ltd supports.

If menu options are customised, then updates to menu items are ignored when an upgrade is performed (this can be avoided by resetting the menu options before performing the upgrade).

Top-level options (menu names in the menu bar) can be:
- created
- renamed
- reordered
- deleted

Options within each of the menus can be:
- moved to be under a different menu name in the menu bar
- reordered under the same menu name in the menu bar
- renamed
- deleted

Changes do not take effect until the next time a Dyalog Session is started.
7.2.8 Windows Tab

Allows customisation of the size and position of floating Edit/Trace windows.

To change the size/position of floating Edit/Trace windows

1. Open the Windows tab.
2. Select the appropriate check boxes and set the variables as required:
   - **Floating windows**
     Toggles whether new Edit and Trace windows are docked inside the Session or floating (undocked). The remaining fields on this tab are only relevant if windows are floating, that is, this option is selected.
   - **Single floating window**
     Select this to display a single floating window with multiple tabs for different Edit/Trace windows. If this option is not selected, then multiple floating windows are displayed for individual Edit/Trace windows.
   - **Remember previous window position**
     Select this to open new floating windows in the positions that were occupied by previously-opened windows when they were closed. The location of the first floating window opened persists between sessions.
   - **Size width and height**
     Specifies the width and height of new Edit and Trace windows.
   - **Posn x and y**
     Specifies the x-y co-ordinates of the screen position for the top left corner of the first Edit/Trace window opened.
   - **Offset x and y**
     Specifies the x-y offsets for the top left corner of Edit/Trace windows opened relative to the previously-opened window. Only relevant if Single floating window is not selected.

7.3 Configuration Parameters

Some customisation can be performed using configuration parameters. These can be specified in the $HOME/.dyalog/dyalog.config file (which is created the first time that Dyalog is run and contains comments describing the format required) or in a shell, but cannot be set from within a Session. For details of the configuration parameters that can be set, and the syntax used to set them, see the Dyalog for macOS Installation and Configuration Guide.
Appendix A  Default Keyboard

The keyboard key mappings shown in figure 7 are the default mappings that are enabled when a Dyalog Session is started with a UK locale.

![Figure 7. The default Session keyboard key mappings (UK keyboard)](image)

To access the glyphs in the lower right quadrant, press ` followed by the appropriate key.

To access the glyphs in the upper right quadrant, press ` followed by the SHIFT key with the appropriate key.

To completely replace the default keyboard with a locale-specific keyboard or to enter Dyalog glyphs in other applications (for example, email), see the Dyalog for macOS Installation and Configuration Guide.
Appendix B   Keyboard Shortcuts

The Dyalog keyboard shortcuts that are supported on macOS are listed in Table 17; those that can be configured in the Shortcuts tab of the Preferences dialog box (see section 7.2.3) are indicated with a * character.

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABT *</td>
<td>About</td>
<td>Shift + F1</td>
<td>Display the About dialog box</td>
</tr>
<tr>
<td>AC *</td>
<td>Align Comments</td>
<td></td>
<td><strong>Edit:</strong> Align comments to current column</td>
</tr>
<tr>
<td>AO *</td>
<td>Comment Out</td>
<td></td>
<td><strong>Session/Edit:</strong> Add comment symbol at start of each tagged or current line</td>
</tr>
<tr>
<td>BH *</td>
<td>Run to exit (in tracer)</td>
<td></td>
<td><strong>Trace:</strong> Continue execution from the current line to completion of the current function or operator. If successful, the selection advances to the next line of the calling function (if there is one).</td>
</tr>
</tbody>
</table>
| BK * | Backward                       | Ctrl + Shift + Backspace | **Session:** Show previous line in input history  
**Edit:** Undo last change (where possible)  
**Trace:** Skip back one line |
| BP * | Toggle Breakpoint              |                    | **Edit:** Toggle a breakpoint on the current line  
**Trace:** Toggle a breakpoint on the current line |
<p>| BT * | Back Tab Window                | Ctrl + Shift + Tab  | Move to previous window in loop                                            |
| CAM *| Clear all trace/stop/monitor   |                    | Remove any trace/stop/monitor flags (as set by ⎕TRACE, ⎕STOP and ⎕MONITOR) from all functions in the workspace |
| CAW *| Close all windows              |                    | Close all open Edit and Trace windows                                       |
| CBP *| Clear stops for active object  |                    | <strong>Edit/Trace:</strong> Clear all breakpoints (resets ⎕STOP) on the function(s).    |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNC *</td>
<td>Connect</td>
<td></td>
<td>Display the RIDE-Dyalog Session dialog box</td>
</tr>
<tr>
<td>CP</td>
<td>Copy</td>
<td>⌘C</td>
<td>Session/Edit: Copy highlighted block of text to the clipboard</td>
</tr>
<tr>
<td>CT</td>
<td>Cut</td>
<td>⌘X</td>
<td>Session/Edit: Delete highlighted block of text and place it on the clipboard</td>
</tr>
<tr>
<td>DB</td>
<td>Backspace</td>
<td>BACKSPACE</td>
<td>Delete character to left of cursor</td>
</tr>
<tr>
<td>DC</td>
<td>Down Cursor</td>
<td>Down</td>
<td>Move cursor down one character</td>
</tr>
<tr>
<td>DI</td>
<td>Delete Item</td>
<td>Fn + Delete</td>
<td>Delete character to right of cursor</td>
</tr>
<tr>
<td>DK *</td>
<td>Delete Block</td>
<td>Delete</td>
<td>Session/Edit: Delete current line (or selected lines if selection exists)</td>
</tr>
</tbody>
</table>
| DL   | Down Limit  | ⌘ + Down Arrow or ⌘ + Fn + Right Arrow | Session: Move cursor to bottom right corner of Session log
<pre><code>                                                                                 | Edit: Move cursor to bottom right corner of content |
</code></pre>
<p>| DMK *| Toggle key display mode |            | Functionality that could be useful when presenting demonstrations.        |
| DMN *| Next line in demo |            | Enables you to display your keystrokes and load/run a demo file.         |
| DMP *| Previous line in demo |            | For more information on presenting demonstrations, enter ] demo -? in a Session. |
| DMR *| Load demo file |            |                                                                              |
| DO * | Uncomment   |            | Session/Edit: Remove comment symbol that is first non-space character on each tagged or current line |
| DS   | Down Screen | Fn + Down Arrow | Move cursor down one screen                                                  |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
</table>
| ED * | Edit    | Shift + Enter | **Session:** Open an **Edit** window (if there is a suspended function on the stack, this opens an **Edit** window for that function – this is called *Naked Edit*)  
**Edit** (not a class or namespace): Open a new **Edit** window for the name under or immediately before/after the cursor.  
**Edit** (a class or namespace): move the cursor to the definition of the name under or immediately before/after the cursor (also see `<JBK>`). |
| EP * | Escape  | Escape      | **Edit:** Fix and Close  
**Trace:** Cut stack back to calling function; close all windows to match new stack status |
| ER * | Enter   | Enter       | **Session:** Execute current line/all modified lines  
**Edit:** Insert new line  
**Trace:** Execute current line |
| EXP *| Expand Selection | Shift + Alt + Up Arrow | Successive presses of `<EXP>` expand the highlighted selection  
**Session:** select the:  
- current line  
- entire Session log  
**Edit:** select the:  
- current token (number, name, string, and so on)  
- current line  
- entire contents of window |
| FD * | Forward | Ctrl + Shift + Enter | **Session:** Show next line in input history  
**Edit:** Reapply last change  
**Trace:** Skip current line |
<p>| FX * | Fix     |             | <strong>Edit:</strong> Fixes the function without closing the <strong>Edit</strong> window |
| HLP *| Help    | F1          | Display the documentation for the system command, system name, control structure keyword or primitive glyph immediately to the left of the cursor |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
</table>
| HO   | Home Cursor | ⌘ + Up Arrow or ⌘ + Fn + Left Arrow | **Session**: Move cursor to top left corner of Session log  
**Edit**: Move cursor to top left corner of content |
<p>| JBK * | Jump Back | Ctrl + Shift + J | <strong>Edit</strong> (a class or namespace): move the cursor back to where the last double-click or <em>Edit</em> command (&lt;ED&gt;) was issued in the current <em>Edit</em> window. Repeatable. |
| JSC * | Show JavaScript Console | F12 | Display the JavaScript console. Only necessary if requested by Dyalog Ltd. when reporting an issue. |
| LBR * | Toggle Language Bar | | Toggle display of the language bar (see section 3.3.2) at the top of the <em>Session</em> window |
| LC   | Left Cursor | Left Arrow | Move cursor left one character |
| LL   | Left Limit | Fn + Left Arrow | Move cursor to the first non-blank character on the current line. If already there, move cursor to start of line. |
| LN * | Toggle Line Numbers | | Turn line numbers on/off in all windows of the same type as the active window. |
| LOG * | Show RIDE Protocol Log | ⌘ + Shift + L | Display the RIDE protocol log. Only necessary if requested by Dyalog Ltd. when reporting an issue. |
| MA * | Continue Execution of All Threads | | Resume execution of any paused threads. For information on threads, see the <em>Dyalog Programming Reference Guide</em>. |
| NEW * | New Session | Ctrl + N | Starts a new Dyalog Session (a new instance of the interpreter) |
| NX * | Next | | <strong>Edit/Trace</strong>: When performing a Search/Replace, locate first match after current one |
| PF1 * | Function Key 1 | | &lt;user-defined functionality&gt; |
| PF2 * - PF10 * | Function Keys 2-10 | F2 – F10 | &lt;user-defined functionality&gt; |
| PF11 * - PF48 * | Function Keys 11-48 | | &lt;user-defined functionality&gt; |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRF *</td>
<td>Show Preferences</td>
<td></td>
<td>Display the Preferences dialog box</td>
</tr>
<tr>
<td>PT</td>
<td>Paste</td>
<td>⌘V</td>
<td>Session: Paste the text contents of the clipboard at cursor Edit: Paste the text contents of the clipboard at cursor</td>
</tr>
<tr>
<td>PV *</td>
<td>Previous match</td>
<td></td>
<td>Edit/Trace: When performing a Search/Replace, locate first match before current one</td>
</tr>
<tr>
<td>QCP</td>
<td>Quick Command Palette</td>
<td></td>
<td>Expose the underlying (Monaco) editor command palette</td>
</tr>
<tr>
<td>QIT *</td>
<td>Quit Session</td>
<td>Ctrl + Q</td>
<td>Terminate the Dyalog Session</td>
</tr>
<tr>
<td>QT *</td>
<td>Quit</td>
<td>Shift + Escape</td>
<td>Session: Undo changes to a previously-entered expression that has not been re-executed and advance the cursor to the next line Edit: Close without saving changes</td>
</tr>
<tr>
<td>RC</td>
<td>Right Cursor</td>
<td>Right Arrow</td>
<td>Move cursor right one character</td>
</tr>
<tr>
<td>RD *</td>
<td>Redraw Function</td>
<td></td>
<td>Edit: Formats function to have correct indentation</td>
</tr>
<tr>
<td>RL</td>
<td>Right Limit</td>
<td>Fn + Right Arrow</td>
<td>Move cursor to the last non-blank character on the current line. If already there, move to end of line.</td>
</tr>
<tr>
<td>RP *</td>
<td>Replace String</td>
<td></td>
<td>Edit: Replace. To do this, enter &lt;RP&gt; and type the string to replace the current search string with (see &lt;SC&gt;); enter &lt;ER&gt; to make the change. Enter &lt;EP&gt; to clear the field.</td>
</tr>
<tr>
<td>SA *</td>
<td>Select All</td>
<td>⌘A</td>
<td>Select all text in the active window</td>
</tr>
<tr>
<td>SBR *</td>
<td>Toggle Status bar</td>
<td></td>
<td>Toggle display of the Status bar (see section 3.3.4) at the bottom of the Session window and floating Edit/Trace windows.</td>
</tr>
<tr>
<td>Code</td>
<td>Command</td>
<td>Keystrokes</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>--------------------</td>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SC *</td>
<td>Search</td>
<td>Ctrl + F</td>
<td><strong>Edit/Trace</strong>: Search. To do this, enter <code>&lt;SC&gt;</code>, type the string to search for and then enter <code>&lt;ER&gt;</code> to find the first occurrence of the string. Enter <code>&lt;EP&gt;</code> to clear the field. Also see related <code>&lt;NX&gt;</code>, <code>&lt;PV&gt;</code>, <code>&lt;RA&gt;</code>, <code>&lt;RP&gt;</code> and <code>&lt;RT&gt;</code>.</td>
</tr>
<tr>
<td>SI *</td>
<td>Strong Interrupt</td>
<td></td>
<td>Suspend code execution as soon as possible (generally after completing execution of the primitive currently being processed)</td>
</tr>
<tr>
<td>STL *</td>
<td>Skip to line</td>
<td></td>
<td><strong>Trace</strong>: Move the current execution marker to the line on which the cursor is positioned</td>
</tr>
<tr>
<td>TB *</td>
<td>Tab Window</td>
<td>Ctrl + Tab</td>
<td>Move to next window in loop</td>
</tr>
<tr>
<td>TC *</td>
<td>Trace</td>
<td>Ctrl + Enter</td>
<td><strong>Session</strong>: If entered directly after an expression, open a Trace window for that expression (<em>explicit trace</em>). If there is a suspended function on the execution stack, open a Trace window for that function (<em>naked trace</em>).</td>
</tr>
<tr>
<td>TGC *</td>
<td>Toggle Comment</td>
<td></td>
<td><strong>Session/Edit</strong>: Toggle a comment glyph at the start of the line in which the cursor is located</td>
</tr>
<tr>
<td>TIP *</td>
<td>Show Value Tips</td>
<td></td>
<td>For the name or glyph under or immediately before the cursor, highlight that name/glyph and display:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• for a name: its referent (for example, the value of a variable or the code of a function).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• for a glyph: the name of the glyph, the keyboard shortcut to enter it, its name and examples of its syntax, arguments and result.</td>
</tr>
<tr>
<td>TL *</td>
<td>Toggle Localisation</td>
<td>Ctrl + Up Arrow</td>
<td><strong>Edit</strong>: For tradfnss, the name under the cursor is added to or removed from the list of localised names on the function’s header line</td>
</tr>
<tr>
<td>Code</td>
<td>Command</td>
<td>Keystrokes</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>TO *</td>
<td>Toggle Outline</td>
<td></td>
<td><strong>Edit</strong>: Open/Close outlined blocks (by default, outlines are shown)</td>
</tr>
</tbody>
</table>