DYALOG

The tool of thought for expert programming

Dyalog for Mac OS User Guide

Version 14.1

Dyalog Limited

Minchens Court, Minchens Lane
Bramley, Hampshire
RG26 5BH
United Kingdom

tel: +44(0)1256 830030
fax: +44 (0)1256 830031
email: support@dyalog.com
http://www.dyalog.com

Version 14.1

Revision: 20151012_141

Contents

1 About This Document

This document is specific to using Dyalog on the Mac OS 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 Mac OS.

1.1 Audience

It is assumed that the reader has a basic knowledge of Mac OS 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).

2 Introduction

The Dyalog development environment offers a user-friendly means of interacting with the interpreter. Support for Dyalog on Mac OS is new at Dyalog version 14.1, and the development environment is not yet as feature-rich as that on Microsoft Windows. However, additional functionality will be added throughout the life of version 14.1 and later.

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 Mac OS.
- Chapter *7* explains how the appearance and behaviour of a Dyalog Session can be customised to meet personal preferences and corporate guidelines.

3 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*

The menu names in the menu bar and the options under each menu can be customised (see section *7.2.6*).

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 Mac OS.

***Table 1**. Dyalog menu options*

Item	Description
About Dyalog	Displays the **About** dialog box, which provides details of the current Dyalog installation.
Preferences	Opens the **Preferences** dialog box (see section *7.2*).
Hide Dyalog	Hides all Dyalog windows and brings the most recently used non-Dyalog application window to the foreground.
Hide Others	Hides the windows of any non-Dyalog applications that are currently running.
Show All	Displays all windows of all currently running applications
Quit Dyalog	Terminates the Dyalog Session (see section *5.9*).

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 2**. File menu options*

Item	Description
New Session	Starts a new Dyalog Session (a new instance of the interpreter).
Connect…	Opens the **RIDE - Connect** 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 - Connect** dialog box, see the *Dyalog RIDE Reference 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*

Item	Description
Undo	Reverses the previous action
Redo	Reverses the effect of the previous Undo
Cut	Deletes the selected text from the active window and places it on the clipboard.
Copy	Copies the selected text to the clipboard.
Paste	Pastes the text contents of the clipboard into the active window at the current location.
Delete	Deletes the selected text without placing it on the clipboard.
Select All	Highlights all characters in the active window.
Start Dictation…	*inserted by Apple – do not use in a Dyalog Session*
Special Characters…	*inserted by Apple – do not use in a Dyalog Session*

A right-click drop-down menu comprising the **Cut**, **Copy** and **Paste** options from the **Edit** menu is available in the **Session** window, all **Edit** windows and the **Trace** window.

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*

Item	Description
Show Language Bar	Toggles display of the language bar (see section *3.3.2*) at the top of the **Session** window.
Float New Editors	Toggles whether new **Edit** or **Trace** windows are docked or floating (see section *5.3*). Does not affect windows that are already open.
Editors on Top	Toggles whether **Edit** windows are always displayed on top of the **Session** window. Only relevant when the **View > Float New Editors** menu option is selected.
Line Wrapping in Session	Toggles use of automatic line wrapping (that is, whether text that exceeds the width of the window wraps to the next line or extends the line with a scroll bar) in the **Session** window.
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	Sets the size of the font in all the windows to its default value.
Theme	Influences the overall style of the user interface. Select from the drop-down list (the initial setting is Cupertino).

3.1.5 Actions Menu

The options available under the **Actions** menu are detailed in Table *5*. These allow currently-running APL code to be interrupted.

***Table 5**. Actions menu options*

Item	Description
Weak Interrupt	Suspends execution of the programme currently being run at the end of the current line.
Strong Interrupt	Suspends execution of the programme currently being run as soon as possible, even if this interrupts the line of code currently being executed.

3.1.6 Window Menu

The options available under the **Window** menu are detailed in Table *6*. These control the display all the Dyalog Session windows.

***Table 6**. Window menu options*

Item	Description
Minimize	Minimises the active window to the dock.
Close Window	Closes: • the active window if **Edit**/**Trace** windows are floating. • the **Session** window if **Edit**/**Trace** windows are docked. If the **Session** window is closed, then all open **Edit** and **Trace** windows are also closed and the Session is terminated.
Bring All to Front	Displays all open Dyalog windows on top of any other open application windows, maintaining their on-screen location, size and layering order.

3.1.7 Help Menu

The options available under the **Help** menu are detailed in Table *7*. These provide access to the Dyalog documentation, website, forum and update portal.

***Table 7**. Help menu options*

Item	Description
Dyalog Help	Opens your default web browser on the welcome page of Dyalog's online help (http://help.dyalog.com).
Documentation Centre	Opens your default web browser on the Documentation Centre page of Dyalog Ltd's website (http://www.dyalog.com/documentation.htm).
Dyalog Website	Opens your default web browser on the home page of Dyalog Ltd's website (http://www.dyalog.com).
MyDyalog	Opens your default web browser on the login page for MyDyalog, the customer portal for updates of Dyalog (https://my.dyalog.com).
Dyalog Forum	Opens your default web browser on the main page of Dyalog Ltd's forum (http://www.dyalog.com/forum).

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 a user's locale and use the appropriate keyboard; if the locale cannot be identified or the locale-specific keyboard has not been configured, then the default configuration is used (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 *7.2.1*).

Alternatively, the default keyboard can be replaced with a locale-specific keyboard in the Session – this locale-specific keyboard also 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 keyboard, see the *Dyalog for Mac OS Installation and Configuration Guide*.

3.3 Session User Interface

The Dyalog Session user interface includes three elements – a caption, a language bar and a **Session** window (as shown in Figure *2*).

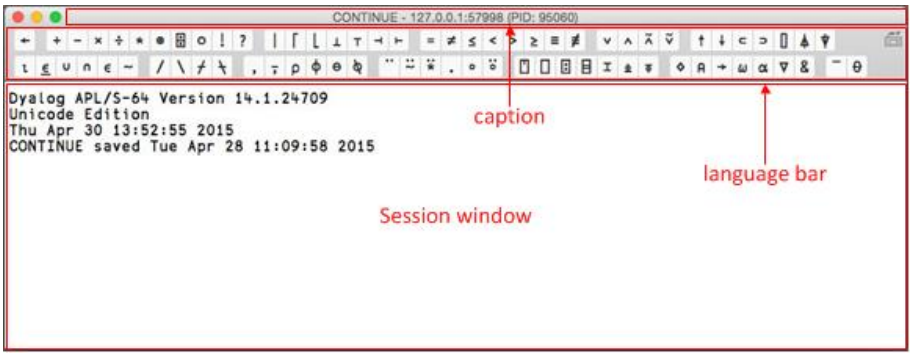


***Figure 2**. The Dyalog Session user interface*

3.3.1 Caption

The caption at the top of the **Session** Window details the ID of the workspace and the host, port and interpreter process numbers.

The caption can be customised (see section *7.2.5*).

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).

On the right hand side of the language bar is the  icon:

- Positioning the cursor over the  icon displays the keyboard shortcuts for command codes – for more information on these, see section *5.1*.
- clicking the mouse on the  icon displays the **Preferences** dialog box (the same as selecting the **Dyalog > Preferences** menu option) – for more information on the **Preferences** dialog box, see section *7.2*.

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

3.3.3 Session Window

The primary purpose of the **Session** window is to provide a scrolling area within which a user 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 Mac OS provides.

4 Input Windows

Instead of just a single **Session** window, 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** window – created and destroyed dynamically as required. There is only one **Trace** window.

Edit and **Trace** windows can be docked or floating (see section *5.3*).

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

The **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 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 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 the cursor over/after `<object name>`

By default, the **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 *3*; the buttons on this toolbar are detailed in table *8*.

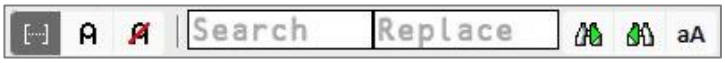


Figure 3. *The **Edit** Window's toolbar*

Table 8. *Buttons on the **Edit** window's toolbar*

Icon	Action	Description
	Toggle line numbers	Toggles line numbers on/off
	Comment selected text	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 selection and at the start of each subsequent line of text within the selection. Same as the *Comment Out* command (**<AO>**).
	Uncomment selected text	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 are not unremoved unless only the comment symbol and subsequent text on that line are selected. Same as the *Uncomment* command (**<DO>**).
	Search and Replace fields	See Section *4.2.2*; the next three buttons relate to the *Search and Replace fields*
	Search for next match	Positions the cursor at the next occurrence of the *Search* text

	Search for previous match	Positions the cursor at the previous occurrence of the *Search* text
aA	Match case	Specifies whether the search is case-sensitive

4.2.2 Search and Replace

The *Search* and *Replace* fields on the **Edit** window's toolbar can be used to locate every occurrence of a specified string (this can include APL glyphs) within the code in the active **Edit** window; optionally, replacement string can be applied on an individual basis.

To search for a string:

1. Enter the string to search for in the *Search* field in one of the following ways:
   - select the string and use the *Search* command (**<SC>**); the selected string is copied to the *Search* field.
   - enter the string directly in the *Search* field – use the *Search* command (**<SC>**) to move the cursor here.

   All occurrences of the specified string are identified 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 highlight the first occurence of the search string after the last position of the cursor.
3. Press either the **Enter** key or the **Search for next match** button  to move the highlighting to the next occurence of the search string.
4. Repeat step 3 as required (the search is cyclic).
5. Use the *Escape* command (**<EP>**) to exit the search functionality.

To replace a string:

1. Enter the string to be replaced in the *Search* field in one of the following ways:
   - select the string and use the *Search* command (**<SC>**); the selected string is copied to the *Search* field.
   - enter the string directly in the *Search* field.

   All occurrences of the specified string are identified 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. Enter the replacement string directly in the *Replace* field.
3. Press the **Enter** key to highlight the first occurence of the search string after the last position of the cursor.

4. Do one of the following:
   - Press the **Enter** key to replace the highlighted occurence of the search string and to move the highlighting to the next occurence of the search string.
   - Press the **Search for next match** button to leave the highlighted occurence of the search string unaltered and move the highlighting to the next occurence of the search string.
5. Repeat step 4 until the required changes have been made (the search is cyclic).
6. Use the *Escape* command (**<EP>**) to exit the search and replace functionality.

4.2.3 Exiting the Edit Window

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

- click on the ✕ icon
- use the *Escape* command (**<EP>**)
- press the **Esc** key

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

- use the *Quit* command (**<QT>**)

4.3 Trace Window

The **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 code 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 `<expression>` **<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 environment variable (for information on environment variables, see the *Dyalog for Mac OS Installation and Configuration Guide*).

By default, the **Trace** window is docked beneath the **Session** and **Edit** windows. Other than setting/removing breakpoints (see section *5.8.1*), **Trace** windows are read-only.

4.3.1 Toolbar

The toolbar displayed at the top of the **Trace** window is shown in figure *4*; the buttons on this toolbar are detailed in table *9*.

Figure 4. *The* ***Trace*** *Window's toolbar*

Table 9*. Buttons on the* ***Trace*** *window's toolbar*

Icon	Action	Description
	Execute line	Executes the current line and advances to the next line
	Trace into expression	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.
	Go back one line	Changes the currently-selected line to be the previous line in the code
	Skip current line	Changes the currently-selected line to be the next line in the code
	Continue trace	Continues execution of the code in the **Trace** window from the current line to completion of the current function or operator, leaving the **Trace** window open. If successful, the code is cut from the stack (its **Trace** window is closed) and the selection advances to the next line of the calling function (if there is one).
	Continue execution	Closes the **Trace** window and resumes execution of the current application thread from the current line.
	Restart all threads	Closes the **Trace** window and resumes execution of all suspended threads.
	Edit name	Opens an **Edit** window for the code in the **Trace** 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.
	Interrupt	Interrupts execution with a weak interrupt
	Clear trace/stop/monitor for this object	Clears all breakpoints (resets ⎕STOP on every function) for the object
	Toggle line numbers	Turns the display of line numbers on/off
	Search field	See Section *4.3.2*; the next three buttons relate to the *Search field*
	Search for next match	Locates the next occurrence of the *Search* text

	Search for previous match	Locates the previous occurrence of the *Search* text
aA	Match case	Specifies whether the search is case-sensitive
✕	Quit this function	Closes the **Trace** window and cuts the code that it contained it from the stack.

4.3.2 Search

The *Search* field on the **Trace** window's toolbar can be used to locate every occurrence of a specified string (this can include APL glyphs) within the code in the **Trace** window.

To search for a string:

1. Enter the string to search for in the *Search* field in one of the following ways:
   - select the string and use the *Search* command (**<SC>**); the selected string is copied to the *Search* field.
   - enter the string directly in the *Search* field – use the *Search* command (**<SC>**) to move the cursor here.

   All occurrences of the specified string are shaded in the **Trace** window. 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 highlight the first occurence of the search string after the last position of the cursor.
3. Press either the **Enter** key or the **Search for next match** button to move the highlighting to the next occurence of the search string.
4. Repeat step 3 as required (the search is cyclic).
5. Use the *Escape* command (**<EP>**) to exit the search functionality.

4.3.3 Exiting the Trace Window

To close the **Trace** window:

- click on the ✕ icon
- 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 using `2023⌶` – see the *Dyalog Language Reference Guide*.

5 Working in a Dyalog Session

The main purpose of a development environment is to enable a user 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  icon in the language bar displays the keyboard shortcuts for command codes.

Keyboard shortcuts can be customised (see section *7.2.2*).

5.2 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>**)
- to discard any changes in the content of the active window before closing it, enter the *Quit* command (**<QT>**)

5.3 Docking and Floating Windows

By default, the **Trace** window and **Edit** windows are docked within the Session. If the menu option **View > Float New Editors** is checked, then any windows created subsequently are not docked but floating.

Changing the setting of **View > Float New Editors** does not affect windows that already exist.

Docked windows can be selected (by clicking within them) and resized (by moving the splitter bar). Floating windows can be selected, moved, resized, maximised and minimised using the standard facilities that Mac OS provides.

The language bar (see section *3.3.2*) is displayed in the **Session** window; it can only be used in the **Session** window and docked **Edit** windows (not in floating **Edit** windows).

5.4 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 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 *5.5.2*) 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:

```
      ⍝ 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
○ `o ``circular
```

```
⌽ `% ``circlestile
⊖ `& ``circlebar
⍉ `^ ``circlebackslash
⍤ `O ``circlediaeresis
```

5.5 Entering Expressions

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

5.5.1 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 **Code** tab of the **Preferences** dialog box (see section *7.2.3*).

5.5.2 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 the user 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 in the **Code** tab of the **Preferences** dialog box (see section *7.2.3*).

5.5.3 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.

5.5.4 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 *8*.

***Table 8.** Syntax colouring convention*

Colour		Syntax
	black	global names
	grey	names namespaces numbers tradfn syntax (specifically, header line and final ∇)
	maroon	control structure keywords
	red	errors (including unmatched parentheses, quotes and braces)
	teal	strings comments
	navy	primitive functions zilde
	blue	idioms (this takes priority over any other syntax colouring) operators parentheses/braces/brackets dfn syntax (specifically, { } ⍺ ⍵ ∇ and :) assignment (←), diamond (⋄) and semi-colon (;)
	purple	system names

Syntax colouring can be customised (see section 7.2.4).

5.6 Executing Expressions

5.6.1 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.

5.6.2 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.

5.6.3 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:
   - 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.
2. Change the expression in some way. The change doesn't 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.

5.7 Threads

Dyalog supports multithreading – the ability to run more than one APL expression at the same time. For information on threads, see the *Dyalog Programmer's Guide*.

5.8 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 *5.8.1*.

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 *5.8.2*.

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 Programmer's Guide*.

5.8.1 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 break point is to be applied or by placing the cursor anywhere in the line before which the break point 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 Language Reference Guide*.

5.8.2 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 Language Reference Guide*.

5.9 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 the red "traffic light" close button

6 Unsupported Language Elements

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

4.1.1 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 the RIDE, ⎕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.

4.1.2 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 F.

However, an operating system restriction means that, in window captions:

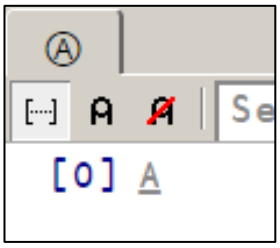
- if the APL385 font is installed, underscored characters are displayed as circled characters
- if the APL385 font is not installed, underscored characters are displayed as ⎕

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):



4.1.3 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 Mac OS. 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 environment variables – 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 environment variables).

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*)
- float/dock new **Edit** and **Trace** windows (see section *5.3*)
- enable/disable automatic line wrapping in **Session** window
- always display floating **Edit** windows on top of the **Session** window
- change the font size in all windows in the Session
- change the overall look of the windows by selecting a pre-defined theme

7.2 Preferences Dialog Box

The **Preferences** dialog box can be used to customise:

- the default keyboard key mappings for APL glyphs (see section *7.2.1*)
- the keyboard shortcuts for command codes (see section *7.2.2*)
- the automatic formatting of text in an **Edit** window (see section *7.2.3*)
- the syntax colouring and background colour (see section *7.2.4*)
- the caption of the **Session** window (see section *7.2.5*)
- the menu options presented by the menu bar (see section *7.2.6*)

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

- selecting the **Dyalog > Preferences** menu option
- entering the *Show Preferences* command (**<PRF>**)
- clicking the  icon on the right hand side of the language bar

7.2.1 Layout 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 Mac OS Installation and Configuration Guide*.

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 **Layout** 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 **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 **Layout** 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 **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.

7.2.2 Shortcuts Tab

Allows customisation of the keyboard shortcuts for command codes (see section *5.1* and Appendix *B*).

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 perfomed.
3. Optionally, delete any existing shortcuts for the command by clicking the cross to the right of the shortcut (this will turn red when moused over).
4. Optionally, add a new shortcut. To do this:
    a. Click the plus symbol to the right of any existing shortcuts (this will turn green when moused over). The **New Shortcut** dialog box will be displayed
    b. In the **New Shortcut** dialog box, enter the keystrokes to map to this action. The **New Shortcut** dialog box 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 **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.

7.2.3 Code Tab

Allows customisation of the automatic formatting/layout of text in an **Edit** window.

To change the way in which text is automatically formatted in the Edit window

1. Open the **Code** tab.
2. Select the appropriate check boxes and set the variables as required:
    - *Auto-indent <number> spaces*
      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.
        i. *in methods: <number> spaces*
           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*
  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 by the user.
- *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.
- *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 *5.5.1*).
- *Auto-close blocks: :If :For ... with <select>*
  Select this to automatically insert matching end statements when an opening control structure statement is entered.
- *Autocompletion after <time> ms*
  Select this and specify a time interval after which the RIDE will display a pop-up window of suggestions based on the characters already entered and the context in which a name is being used (see section *5.5.2*).
- *Code folding (outlining)*
  Select this to enable code folding/outlining of control structures (including `:Section` structures) and functions. When changes to outlining are applied, existing code in an open **Edit** window is automatically updated to relect the new rules.

3. **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.

7.2.4 Colours Tab

Allows customisation of the syntax colouring (see section *5.5.4*). 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 **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 **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 check boxes 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 **Rename**.
      The name in the Scheme field becomes editable.
   b. Edit the name in the **Scheme** field.
6. Click **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 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.
3. Click **OK**.

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

The default setting of: `{WSID} – {HOST} : {PORT} (PID: {PID})` gives a different result on every machine. For example:

```
CONTINUE – 127.0.0.1:49376 (PID: 293)
```

Changing this to `{CHARS} {BITS}` gives information that could be the same on multiple machines:

```
Unicode 64
```

7.2.6 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 updating the installed version of RIDE (this can be avoided by resetting the menu options before upgrading RIDE).

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.

7.3 Environment Variables

Some customisation can be performed using environment variables. 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 environment variables that can be set, and the syntax used to set them, see the *Dyalog for Mac OS Installation and Configuration Guide*.

Appendix A Default Keyboard

The keyboard key mappings shown in figure *1* are the default mappings enabled when a Dyalog Session is started with a UK locale.

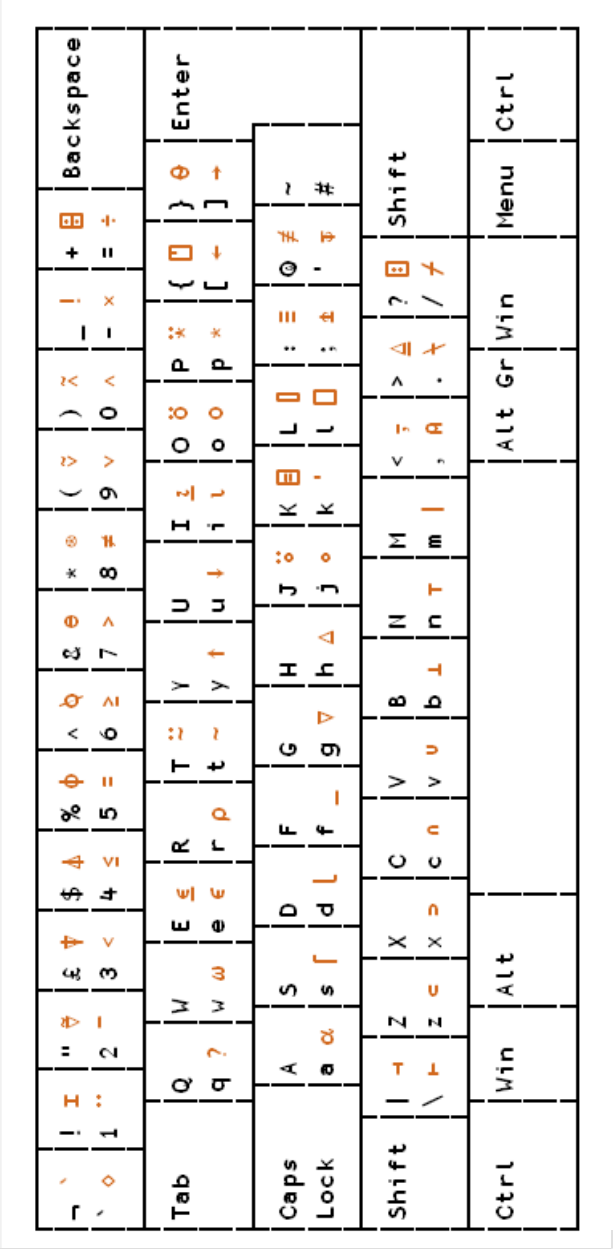


***Figure 1**. The default Session keyboard key mappings (shown on a UK keyboard).*

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.

Appendix B Keyboard Shortcuts

The Dyalog keyboard shortcuts that are supported on Mac OS are listed in *Table 1*; those that can be configured in the **Shortcuts** tab of the **Preferences** dialog box (see section *7.2.2*) are indicated with a * character. Keyboard shortcuts that are not currently supported on Mac OS are detailed in *Table 2*.

***Table 1**. Dyalog keyboard shortcuts supported on Mac OS*

Keycode	Command	Keystrokes	Description
ABT *	About	Shift + F1	Display the **About** dialog box
AC *	Align Comments		**Edit**: Align comments to current column
AO *	Comment Out		**Edit**: Add comment symbol at start of each tagged or current line
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
BT *	Back Tab Window	Ctrl + Shift + Tab	Move to previous window in loop
CNC *	Connect		Display the **RIDE - Connect** dialog box
CP	Copy	⌘ + C	**Session/Edit**: Copy highlighted block of text to the clipboard
CT	Cut	⌘ + X	**Session/Edit**: Delete highlighted block of text and place it on the clipboard
DB	Backspace	Backspace	Delete character to left of cursor
DC	Down Cursor	Down	Move cursor down one character
DI	Delete Item	Fn + Delete	Delete character to right of cursor

DK	Delete Block	Delete	**Session/Edit**: Delete highlighted block of text
DL	Down Limit	⌘ + Down Arrow or ⌘ + Fn + Right Arrow	**Session**: Move cursor to bottom right corner of Session log **Edit**: Move cursor to bottom right corner of content
DMK	Toggle key display mode		Functionality that could be useful when presenting demonstrations. Enable you to display your keystrokes and load/run a demo file.
DMN	Next line in demo		
DMP	Previous line in demo		
DMR	Load demo file		
DO *	Uncomment		**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
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*)
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
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
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
IN *	Insert Mode		Toggle between insert and overwrite modes
LBR *	Toggle Language Bar		Toggle display of the language bar (see section *3.3.2*) at the top of the **Session** window
LC	Left Cursor	Left Arrow	Move cursor left one character
LL	Left Limit	Fn + Left Arrow	Move cursor left as far as possible in current row
LN *	Toggle Line Numbers		Turn line numbers on/off in all windows of the same type as the active window.
NEW *	New Session	Ctrl + N	Starts a new Dyalog Session (a new instance of the interpreter)
PRF *	Show Preferences		Display the **Preferences** dialog box
PT	Paste	⌘ + V	**Session**: Paste the text contents of the clipboard at cursor **Edit**: Paste the text contents of the clipboard at cursor
QIT *	Quit Session	Ctrl + Q	Terminate the Dyalog Session
QT *	Quit	Shift + Escape	**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
RC	Right Cursor	Right Arrow	Move cursor right one character
RD *	Redraw Function		**Edit**: Formats function to have correct indentation

RL	Right Limit	Fn + Right Arrow	Move cursor right as far as possible in current row
RP *	Replace String	Ctrl + G	**Edit**: Replace. To do this, enter **<RP>** and type the string to replace the current search string with (see **<SC>**); enter **<ER>** to make the change. Enter **<EP>** to clear the field.
SC *	Search	Ctrl + F	**Edit/Trace**: Search. To do this, enter **<SC>**, type the string to search for and then enter **<ER>** to find the first occurrence of the string. Enter **<EP>** to clear the field. Also see related **<NX>**, **<PV>**, **<RA>**, **<RP>** and **<RT>**.
SI *	Strong Interrupt		Suspend code execution as soon as possible (generally after completing execution of the primitive currently being processed)
TB *	Tab Window	Ctrl + Tab	Move to next window in loop
TC *	Trace	Ctrl + Enter	**Session**: If there is a suspended function on the stack, open a **Trace** window for that function (*Naked Trace*)
TL *	Toggle Localisation	Ctrl + Up Arrow	**Edit**: For tradfns, the name under the cursor is added to or removed from the list of localised names on the function's header line
TO *	Toggle Outline		**Edit**: Open/Close outlined blocks (by default, outlines are shown)
UC	Up Cursor	Up Arrow	Move cursor up one character
UL	Up Limit	⌘ + 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
US	Up Screen	Fn + Up Arrow	Move cursor up one screen
WI *	Weak Interrupt		Suspend code execution at the start of the next line (generally after completing execution of the statement currently being processed)
ZMI *	Increase Font Size	Ctrl + =	Increase the size of the font in all the windows

ZMO *	Decrease Font Size	Ctrl + -	Decrease the size of the font in all the windows
ZMR *	Reset Font Size	Ctrl + 0	Reset the size of the font in all the windows to its default value

***Table 2**. Dyalog keyboard shortcuts not currently supported on Mac OS*

Keycode	Command	Description
BH	Run To Exit	**Trace**: Continue execution but show **Trace** window or stop on next error
CB	Clear Breakpoints	**Edit**: Clears all breakpoints from the current object **Trace**: Clears all breakpoints from the current object
FX	Fix	**Edit**: Fixes the function without closing the **Edit** window
HK	Hot Key (□SM)	□SM: With non-empty □SM, toggle between □SM window and **Trace**/**Edit**/**Session** window
JP	Jump	Toggle between **Session** window and current window
LS	Left Screen	Move cursor left one screen
MO	Move To Outline	**Edit**: When on the first or last line of a control structure, move to the opposite end (by default, outlines are not shown)
MR	Move/Resize	Move and/or resize the current window. To do this, enter **<MR>** then: • **<LC>**/**<RC>**/**<UC>**/**<DC>**: move window in specified direction • **<LS>**/**<RS>**/**<US>**/**<DS>**: move bottom right corner of window in selected direction relative to top left corner • **<EP>**: exit move/resize mode
MV	Move Block	**Session**: Move highlighted block of text to below the current line
NX	Next	**Edit/Trace**: When performing a Search/Replace, locate first match after current one
OP	Open Line	**Edit**: Opens a line underneath the current line
PV	Previous	**Edit/Trace**: When performing a Search/Replace, locate first match before current one
RA	Repeat All	When performing a Search/Replace, perform the same action again in all directions (Caution: replaces all matches in current object)
RM	Resume Execution	**Trace**: Continue execution and do not show **Trace** window or stop on next error
RS	Right Screen	Move cursor right one screen

RT	Repeat (Do)	When performing a Search/Replace, perform the same action again
SR	Screen Refresh	Redraw the session, removing all extraneous text that has come from external sources and resetting the session display
TG	Tag	**Session**: Tag (highlight) blocks of text. To do this, position the cursor at the start of the text to tag, enter **<TG>**, move to the end of the text to tag and enter **<TG>** again. Enter **<TG>** a third time to clear the current tagging.
ZM	Zoom	Toggle current window between current size and full screen size