Dyalog for macOS
UI Guide
Version 16.0

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
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 ............................................................................... 8
    3.1.3 Edit Menu .............................................................................. 8
    3.1.4 View Menu ............................................................................. 9
    3.1.5 Window Menu ......................................................................... 9
    3.1.6 Action Menu .......................................................................... 9
    3.1.7 Help Menu ............................................................................ 10
  3.2 Keyboard Key Mappings for APL Glyphs ...................................... 10
  3.3 Session User Interface ................................................................. 10
    3.3.1 Caption ................................................................................ 11
    3.3.2 Language Bar .......................................................................... 11
    3.3.3 Tabbed Area .......................................................................... 11
    3.3.4 Session Window ..................................................................... 12
    3.3.5 Workspace Explorer ............................................................. 12
    3.3.6 Debug Information Window .................................................. 12

4 INPUT WINDOWS ...................................................................... 14
  4.1 Session Window .......................................................................... 14
  4.2 Edit Window ................................................................................. 14
    4.2.1 Toolbar .................................................................................. 15
    4.2.2 Search and Replace ................................................................. 16
    4.2.3 Exiting the Edit Window .......................................................... 17
  4.3 Trace Window .............................................................................. 17
    4.3.1 Toolbar .................................................................................. 18
    4.3.2 Search ................................................................................... 19
    4.3.3 Exiting the Trace Window ....................................................... 19

5 WORKING IN A DYALOG SESSION ........................................ 20
  5.1 Keyboard Shortcuts and Command Codes .................................. 20
  5.2 Accented Characters .................................................................. 20
  5.3 Navigating the Windows ............................................................. 20
  5.4 Display of Windows .................................................................... 21
  5.5 Entering APL Characters ............................................................ 21
  5.6 Entering Expressions ................................................................. 22
    5.6.1 Paired Enclosures ................................................................. 22
    5.6.2 Autocomplete ........................................................................ 22
    5.6.3 Context-Sensitive Help ......................................................... 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6.4</td>
<td>Syntax Colouring</td>
<td>23</td>
</tr>
<tr>
<td>5.7</td>
<td>Executing Expressions</td>
<td>24</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Executing a New Expression</td>
<td>24</td>
</tr>
<tr>
<td>5.7.2</td>
<td>Re-executing a Previous Expression</td>
<td>24</td>
</tr>
<tr>
<td>5.7.3</td>
<td>Re-executing Multiple Previous Expressions</td>
<td>24</td>
</tr>
<tr>
<td>5.8</td>
<td>Threads</td>
<td>25</td>
</tr>
<tr>
<td>5.9</td>
<td>Suspending Execution</td>
<td>25</td>
</tr>
<tr>
<td>5.9.1</td>
<td>Breakpoints</td>
<td>25</td>
</tr>
<tr>
<td>5.9.2</td>
<td>Interrupts</td>
<td>26</td>
</tr>
<tr>
<td>5.10</td>
<td>Terminating a Session</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>UNSUPPORTED LANGUAGE ELEMENTS</td>
<td>27</td>
</tr>
<tr>
<td>6.1</td>
<td>Function Key Configuration</td>
<td>27</td>
</tr>
<tr>
<td>6.2</td>
<td>Underscored Characters in Window Captions</td>
<td>27</td>
</tr>
<tr>
<td>6.3</td>
<td>Operating System Terminal Window Interaction</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>CUSTOMISING YOUR SESSION</td>
<td>28</td>
</tr>
<tr>
<td>7.1</td>
<td>View Menu</td>
<td>28</td>
</tr>
<tr>
<td>7.2</td>
<td>Preferences Dialog Box</td>
<td>28</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Layout Tab</td>
<td>29</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Shortcuts Tab</td>
<td>30</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Code Tab</td>
<td>30</td>
</tr>
<tr>
<td>7.2.4</td>
<td>Colours Tab</td>
<td>32</td>
</tr>
<tr>
<td>7.2.5</td>
<td>Title Tab</td>
<td>33</td>
</tr>
<tr>
<td>7.2.6</td>
<td>Menu Tab</td>
<td>33</td>
</tr>
<tr>
<td>7.3</td>
<td>Environment Variables</td>
<td>34</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>default keyboard</td>
<td>35</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>keyboard shortcuts</td>
<td>36</td>
</tr>
</tbody>
</table>
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 Code tab of the Preferences dialog box, as detailed in section 7.2.3).
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.

![Menu bar menu options](image)

**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 macOS.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Dyalog</td>
<td>Displays the <em>About</em> dialog box, which provides details of the current Dyalog installation.</td>
</tr>
<tr>
<td>Preferences</td>
<td>Opens the <em>Preferences</em> 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>New Session</td>
<td>Starts a new Dyalog Session (a new instance of the interpreter).</td>
</tr>
<tr>
<td>Connect...</td>
<td>Opens the <em>RIDE - Connect</em> dialog box, enabling you to connect to specified or remote RIDE-enabled Dyalog process. For information on RIDE, connecting to remote interpreters and the <em>RIDE - Connect</em> dialog box, see the <em>RIDE User Guide</em>.</td>
</tr>
</tbody>
</table>
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 ⌘X</td>
<td>Deletes the selected text from the active window and places it on the clipboard.</td>
</tr>
<tr>
<td>Copy ⌘C</td>
<td>Copies the selected text to the clipboard.</td>
</tr>
<tr>
<td>Paste ⌘V</td>
<td>Pastes the text contents of the clipboard into the active window at the current location.</td>
</tr>
<tr>
<td>Select All ⌘A</td>
<td>Highlights all characters in the active window.</td>
</tr>
<tr>
<td>Start Dictation…</td>
<td>inserted by Apple – not supported by Dyalog Ltd</td>
</tr>
<tr>
<td>Emoji &amp; Symbols…</td>
<td>inserted by Apple – not supported by Dyalog Ltd</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 Session 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 Session window.</td>
</tr>
<tr>
<td>Show Debug</td>
<td>Toggles display of the debug information window (see section 3.3.6) to the right of the Session window.</td>
</tr>
<tr>
<td>Line Wrapping in Session</td>
<td>Toggles line wrapping. When off, ⎕PW is used to determine line length. When on, a combination of ⎕PW and line wrapping is used.</td>
</tr>
<tr>
<td>Increase Font Size ⌘=</td>
<td>Increases the size of the font in all the windows</td>
</tr>
<tr>
<td>Decrease Font Size ⌘-</td>
<td>Decreases the size of the font in all the windows</td>
</tr>
<tr>
<td>Reset Font Size ⌘0</td>
<td>Sets the size of the font in all the windows to its default value.</td>
</tr>
<tr>
<td>Enter Full Screen ⌘f</td>
<td>Toggles the entire session between its current size and full screen size.</td>
</tr>
</tbody>
</table>

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>
</tbody>
</table>

3.1.6 Action Menu

The options available under the Actions 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>Edit</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>Trace</td>
<td>If selected when the cursor is positioned directly after an expression, open a Trace window for that expression (explicit trace). If there is a suspended function on the execution stack, open a Trace window for that function (naked trace).</td>
</tr>
<tr>
<td>Weak Interrupt</td>
<td>Suspends execution of the programme currently being run at the start of the next line.</td>
</tr>
<tr>
<td>Strong Interrupt</td>
<td>Suspends execution of the programme currently being run as soon as possible, even if this interrupts the line of code currently being executed.</td>
</tr>
</tbody>
</table>

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.

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

Table 7. Help menu options

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dyalog Help</td>
<td>Opens your default web browser on the welcome page of Dyalog's online help (<a href="http://help.dyalog.com">http://help.dyalog.com</a>).</td>
</tr>
<tr>
<td>Documentation Centre</td>
<td>Opens your default web browser on the Documentation Centre page of Dyalog Ltd’s website (<a href="http://www.dyalog.com/documentation.htm">http://www.dyalog.com/documentation.htm</a>).</td>
</tr>
<tr>
<td>Dyalog Website</td>
<td>Opens your default web browser on the home page of Dyalog Ltd’s website (<a href="http://www.dyalog.com">http://www.dyalog.com</a>).</td>
</tr>
<tr>
<td>MyDyalog</td>
<td>Opens your default web browser on the login page for MyDyalog, the customer portal for updates of Dyalog (<a href="https://my.dyalog.com">https://my.dyalog.com</a>).</td>
</tr>
<tr>
<td>Dyalog Forum</td>
<td>Opens your default web browser on the main page of Dyalog Ltd’s forum (<a href="http://www.dyalog.com/forum">http://www.dyalog.com/forum</a>).</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 a user’s locale and use the appropriate key mappings; if the locale cannot be identified or the locale-specific key mappings have 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 7.2.1).

Alternatively, the default set of key mappings can be replaced with a different set of key mappings in the Session – this 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 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 tabbed area (see section 3.3.3)
- a Session window (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)
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.

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 (<LBR>).

3.3.3 Tabbed Area

The tabbed area is located at the top of the Session window, beneath the language bar. By default it contains tabs for the Session window and any open workspace explorer, debug information window and Edit windows.
All windows can be moved to different locations (see section 5.4) – doing this impacts the locations of their tabs.

### 3.3.4 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 macOS provides.

### 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 4; the icons on this toolbar are detailed in table 8.

![Threads area's toolbar](image)

**Figure 4. The Threads area’s toolbar**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Refresh" /></td>
<td>Refresh now</td>
<td>Refresh the list of threads</td>
</tr>
<tr>
<td><img src="image" alt="Continue" /></td>
<td>Continue execution of this thread</td>
<td>Resume execution of the currently selected thread.</td>
</tr>
<tr>
<td><img src="image" alt="Continue all" /></td>
<td>Continue execution of all threads</td>
<td>Resume execution of any paused threads. For information on threads, see the <em>Dyalog Programming Reference Guide</em>.</td>
</tr>
</tbody>
</table>
• The SI Stack area lists the functions in the execution stack; each function in the list also has a line number indicating 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 can be multiple Trace windows.

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 RIDE’s 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 (\texttt{<ED>}), double-clicking the cursor or clicking the \(\text{EDIT}\) 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 \texttt{<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 \textit{naked edit}.

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

- Move the cursor over/after \texttt{<object name>} and enter the Edit command (\texttt{<ED>})
- Double-click with the cursor over/after \texttt{<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 5; the buttons on this toolbar are detailed in table 9.

![Figure 5. The Edit Window’s toolbar](image)

<table>
<thead>
<tr>
<th>Table 9. Icons on the Edit window’s toolbar</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Icon</strong></td>
</tr>
<tr>
<td>[·:]</td>
</tr>
<tr>
<td>✎</td>
</tr>
<tr>
<td>📈</td>
</tr>
<tr>
<td>⋆</td>
</tr>
<tr>
<td>🔍</td>
</tr>
</tbody>
</table>
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, a 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:
   - Press the Search button and enter the string directly in the Search field.
   - Enter the Search command (\textless SC\textgreater) and enter the string directly in the Search field.
   - Select the string in the Edit window and enter the Search command (\textless SC\textgreater) or press the Search button; the selected string is copied to the Search 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 3 as required (the search is cyclic).

5. Use the Escape command (\textless EP\textgreater) or the Esc key to exit the search functionality.

To replace a string:

1. Enter the string to be replaced in the Replace field in one of the following ways:
   - Press the Replace button and enter the string directly in the Replace field.
   - Enter the Replace command (\textless RP\textgreater) and enter the string directly in the Replace field.
   - Select the string in the Edit window and enter the Replace command (\textless RP\textgreater) or press the Replace button; the selected string is copied to the Replace field.

   Press the Enter key – the Replace field is replaced by a With field.
2. Enter the replacement string directly in the With 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 click YES to replace the selected occurrence of the search string and to advance the selection to the next occurrence of the search string.
   - Click NO to leave the selected occurrence of the search string unaltered and to advance the selection to the next occurrence of the search string.
   - Click ALL to replace all occurrences of the search string.

5. Click STOP, use the Escape command (<EP>) or press the Esc key 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

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 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 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 6; the buttons on this toolbar are detailed in table 10.

![Figure 6. 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="Toggle 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="Execute line" /></td>
<td>Execute line</td>
<td>Executes the current line and advances to the next line.</td>
</tr>
<tr>
<td><img src="image" alt="Trace into expression" /></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="Go back one line" /></td>
<td>Go back one line</td>
<td>Changes the currently-selected line to be the previous line in the code.</td>
</tr>
<tr>
<td><img src="image" alt="Skip current line" /></td>
<td>Skip current line</td>
<td>Changes the currently-selected line to be the next line in the code.</td>
</tr>
<tr>
<td><img src="image" alt="Stop on next line of calling function" /></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="Continue execution of this thread" /></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="Continue execution of all threads" /></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="Edit name" /></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="Interrupt" /></td>
<td>Interrupt</td>
<td>Interrupts execution with a weak interrupt.</td>
</tr>
<tr>
<td><img src="image" alt="Clear trace/stop/monitor for this object" /></td>
<td>Clear trace/stop/monitor for this object</td>
<td>Clears all breakpoints (resets ⎕STOP on every function) for the object.</td>
</tr>
<tr>
<td><img src="image" alt="Search and Replace fields" /></td>
<td>Search and Replace fields</td>
<td>See Section 4.3.2; the next three buttons relate to the Search field.</td>
</tr>
<tr>
<td><img src="image" alt="Search" /></td>
<td>Search</td>
<td>Opens the Search field, enabling a search to be performed.</td>
</tr>
<tr>
<td><img src="image" alt="Search for previous match" /></td>
<td>Search for previous match</td>
<td>Positions the cursor at the previous occurrence of the Search text.</td>
</tr>
<tr>
<td><img src="image" alt="Search for next match" /></td>
<td>Search for next match</td>
<td>Positions the cursor at the next occurrence of the Search text.</td>
</tr>
</tbody>
</table>
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:
   - Press the Search button  and enter the string directly in the Search field.
   - Enter the Search command (<SC>) and enter the string directly in the Search field.
   - Select the string in the Trace window and enter the Search command (<SC>) or press the Search button ; the selected string is copied to the Search field.

   All occurrences of the specified string are highlighted 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 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 3 as required (the search is cyclic).

5. Use the Escape command (<EP>) or the Esc key to exit the search functionality.

4.3.3 Exiting the Trace Window

To close the Trace window:

- click on the X 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 APL 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 a selection of the keyboard shortcuts for command codes.

Keyboard shortcuts can be customised (see section 7.2.2).

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, ā, ä, æ 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>)
- 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 (by toggling the icon at the top right of each window).

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 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:

```
⍝ 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
ϕ `% `circletile
œ `& `circlebar
♀ `^ `circlebackslash
♀O `O `circlediachesis

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

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 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).

5.6.3 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 11.

<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>
<tr>
<td>blue</td>
<td>idioms (this takes priority over any other syntax colouring)</td>
</tr>
<tr>
<td></td>
<td>operators</td>
</tr>
<tr>
<td></td>
<td>parentheses/braces/brackets</td>
</tr>
<tr>
<td></td>
<td>dfn syntax (specifically, { } ⋄ ⋄ ⋄ ⋄ and :)</td>
</tr>
<tr>
<td></td>
<td>assignment ⋄, diamond ⋄ and semi-colon ⋄</td>
</tr>
<tr>
<td>purple</td>
<td>system names</td>
</tr>
</tbody>
</table>

Syntax colouring can be customised (see section 7.2.4).
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 (\texttt{<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 (\texttt{<BK>}) and the Forward command (\texttt{<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 (\texttt{<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 (\texttt{<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, \texttt{)LOAD} and \texttt{)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 (\texttt{<BK>}) and the Forward command (\texttt{<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 (\texttt{<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.

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 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 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. Instead, function keys should be set through the Shortcuts tab of the Preferences dialog box (see Section 7.2.2).

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 ₒ. 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):

![Edit window with underscored name]

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 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)
- 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

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 code 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 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 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 performed.
3. Optionally, delete any existing shortcuts for the command by clicking the red cross to the right of the shortcut.
4. Optionally, add a new shortcut. To do this:
   a. Click 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 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 checkboxes 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 ᾱ glyph. If this is not selected, then lines that start with the ᾱ 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.

- **Handle formatting through the interpreter**
  Select this to delegate all formatting of code to the interpreter.

- **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).

- **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 5.6.2).

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

- **Blink cursor**
  Select this to make the cursor blink rather than be displayed constantly.

- **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 reflect the new rules.

- **Show value tips**
  Select this to display the referent of a name. When the cursor is positioned over or immediately before 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 quit prompt**
  Select this to display a confirmation dialog box when exiting the RIDE session.

- **Show tips for glyphs**
  Select this to show the tooltip for a glyph. When the cursor is
positioned over or immediately before 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.

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 5.6.4). 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 **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 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 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 – clicking on these inserts them into the Window title field.

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.

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

7.2.6 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 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.

![Keyboard Image]

Figure 7. The default Session keyboard key mappings (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.

⚠️ 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 12; those that can be configured in the Shortcuts tab of the Preferences dialog box (see section 7.2.2) are indicated with a * character.

Table 12. Dyalog keyboard shortcuts supported on macOS

<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>Edit: Align comments to current column</td>
</tr>
<tr>
<td>AO *</td>
<td>Comment Out</td>
<td></td>
<td>Session/Edit: Add comment symbol at start of each tagged or current line</td>
</tr>
<tr>
<td>BK *</td>
<td>Backward</td>
<td>Ctrl + Shift + Backspace</td>
<td>Session: Show previous line in input history Edit: Undo last change (where possible) Trace: Skip back one line</td>
</tr>
<tr>
<td>BP *</td>
<td>Toggle Breakpoint</td>
<td></td>
<td>Edit: Toggle a breakpoint on the current line Trace: Toggle a breakpoint on the current line</td>
</tr>
<tr>
<td>BT *</td>
<td>Back Tab Window</td>
<td>Ctrl + Shift + Tab</td>
<td>Move to previous window in loop</td>
</tr>
<tr>
<td>CNC +</td>
<td>Connect</td>
<td></td>
<td>Display the RIDE - Connect 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 highlighted block of text</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>DL</td>
<td>Down Limit</td>
<td>$\mathcal{M}$ + Down Arrow or $\mathcal{M}$ + Fn + Right Arrow</td>
<td><strong>Session:</strong> Move cursor to bottom right corner of Session log&lt;br&gt;<strong>Edit:</strong> Move cursor to bottom right corner of content</td>
</tr>
<tr>
<td>DMK *</td>
<td>Toggle key display mode</td>
<td></td>
<td>Functionality that could be useful when presenting demonstrations.</td>
</tr>
<tr>
<td>DMN *</td>
<td>Next line in demo</td>
<td></td>
<td>Enables you to display your keystrokes and load/run a demo file.</td>
</tr>
<tr>
<td>DMP *</td>
<td>Previous line in demo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMR *</td>
<td>Load demo file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DO *</td>
<td>Uncomment</td>
<td></td>
<td><strong>Session/Edit:</strong> Remove comment symbol that is first non-space character on each tagged or current line</td>
</tr>
<tr>
<td>DS</td>
<td>Down Screen</td>
<td>Fn + Down Arrow</td>
<td>Move cursor down one screen</td>
</tr>
<tr>
<td>ED *</td>
<td>Edit</td>
<td>Shift + Enter</td>
<td><strong>Session:</strong> Open an <strong>Edit</strong> window (if there is a suspended function on the stack, this opens an <strong>Edit</strong> window for that function – this is called <strong>Naked Edit</strong>)&lt;br&gt;<strong>Edit</strong> (not a class or namespace): Open a new <strong>Edit</strong> window for the name under or immediately before/after the cursor.&lt;br&gt;<strong>Edit</strong> (a class or namespace): move the cursor to the definition of the name under or immediately before/after the cursor (also see $&lt;$JBK$&gt;$).</td>
</tr>
<tr>
<td>EMD *</td>
<td>Edit mode</td>
<td>Shift + F3</td>
<td><strong>Trace:</strong> Open an <strong>Edit</strong> window for the contents of the active <strong>Trace</strong> window irrespective of cursor position within that window.</td>
</tr>
<tr>
<td>EP *</td>
<td>Escape</td>
<td>Escape</td>
<td><strong>Edit:</strong> Fix and Close&lt;br&gt;<strong>Trace:</strong> Cut stack back to calling function; close all windows to match new stack status</td>
</tr>
<tr>
<td>ER *</td>
<td>Enter</td>
<td>Enter</td>
<td><strong>Session:</strong> Execute current line/all modified lines&lt;br&gt;<strong>Edit:</strong> Insert new line&lt;br&gt;<strong>Trace:</strong> Execute current 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>
</tbody>
</table>
| EXP | Expand | 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 |
| FX | Fix |  | **Edit:** Fixes the function without closing the Edit 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 |
| 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 Edit command (&lt;ED&gt;) was issued in the current Edit 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 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. |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG *</td>
<td>Show RIDE Protocol Log</td>
<td>⌘ + Shift + L</td>
<td>Display the RIDE protocol log. Only necessary if requested by Dyalog Ltd. when reporting an issue.</td>
</tr>
<tr>
<td>MA *</td>
<td>Continue Execution of All Threads</td>
<td></td>
<td>Resume execution of any paused threads. For information on threads, see the <em>Dyalog Programming Reference Guide</em>.</td>
</tr>
<tr>
<td>NEW *</td>
<td>New Session</td>
<td>Ctrl + N</td>
<td>Starts a new Dyalog Session (a new instance of the interpreter)</td>
</tr>
<tr>
<td>NX *</td>
<td>Next</td>
<td></td>
<td><em>Edit/Trace:</em> When performing a Search/Replace, locate first match after current one</td>
</tr>
<tr>
<td>PF1 *</td>
<td>Function Key 1</td>
<td></td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF2 *</td>
<td>Function Key 2</td>
<td>F2</td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF3 *</td>
<td>Function Key 3</td>
<td>F3</td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF4 *</td>
<td>Function Key 4</td>
<td>F4</td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF5 *</td>
<td>Function Key 5</td>
<td>F5</td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF6 *</td>
<td>Function Key 6</td>
<td>F6</td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF7 *</td>
<td>Function Key 7</td>
<td>F7</td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF8 *</td>
<td>Function Key 8</td>
<td>F8</td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF9 *</td>
<td>Function Key 9</td>
<td>F9</td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF10 *</td>
<td>Function Key 10</td>
<td>F10</td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF11 *</td>
<td>Function Key 11</td>
<td></td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PF12 *</td>
<td>Function Key 12</td>
<td></td>
<td><em>&lt;user defined functionality&gt;</em></td>
</tr>
<tr>
<td>PRF *</td>
<td>Show Preferences</td>
<td></td>
<td>Display the <em>Preferences</em> dialog box</td>
</tr>
</tbody>
</table>
| PT | Paste | ⌘ V | *Session:* Paste the text contents of the clipboard at cursor  
*Edit:* Paste the text contents of the clipboard at cursor |
<p>| PV * | Previous match | | <em>Edit/Trace:</em> When performing a Search/Replace, locate first match before current one |
| QIT * | Quit Session | Ctrl + Q | Terminate the Dyalog Session |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 |             | **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>`. |
<p>| 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 | <strong>Session:</strong> If entered directly after an expression, open a Trace window for that expression (explicit trace). If there is a suspended function on the execution stack, open a Trace window for that function (naked trace). |
| TGC *| Toggle Comment |             | <strong>Session/Edit:</strong> Toggle a comment glyph at the start of the line in which the cursor is located |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Keystrokes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 tradfns, 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>TO *</td>
<td>Toggle Outline</td>
<td></td>
<td><strong>Edit:</strong> Open/Close outlined blocks (by default, outlines are shown)</td>
</tr>
<tr>
<td>TR *</td>
<td>Refresh Threads</td>
<td></td>
<td><strong>Debug information window, Threads area:</strong> Refresh the list of threads</td>
</tr>
<tr>
<td>UC</td>
<td>Up Cursor</td>
<td>Up Arrow</td>
<td>Move cursor up one character</td>
</tr>
<tr>
<td>UL</td>
<td>Up Limit</td>
<td>⌘ + Up Arrow or ⌘ + Fn + Left Arrow</td>
<td><strong>Session:</strong> Move cursor to top left corner of Session log</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Edit:</strong> Move cursor to top left corner of content</td>
</tr>
<tr>
<td>US</td>
<td>Up Screen</td>
<td>Fn + Up Arrow</td>
<td>Move cursor up one screen</td>
</tr>
<tr>
<td>VAL *</td>
<td>Evaluate Selection or Name</td>
<td></td>
<td><strong>Edit:</strong> Evaluate the selected expression or name and display the result in the Session window.</td>
</tr>
<tr>
<td>WI *</td>
<td>Weak Interrupt</td>
<td>Ctrl + Pause</td>
<td>Suspend code execution at the start of the next line (generally after completing execution of the statement currently being processed)</td>
</tr>
<tr>
<td>WSE *</td>
<td>Toggle Workspace Explorer</td>
<td></td>
<td>Toggle display of the workspace explorer (see section 3.3.5) to the left of the Session window</td>
</tr>
<tr>
<td>ZM *</td>
<td>Toggle maximise Edit window</td>
<td></td>
<td>Toggle active Edit window between current size and full Session size</td>
</tr>
<tr>
<td>ZMI *</td>
<td>Increase Font Size</td>
<td>Ctrl + =</td>
<td>Increase the size of the font in all the 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>ZMO *</td>
<td>Decrease Font Size</td>
<td>Ctrl + -</td>
<td>Decrease the size of the font in all the windows</td>
</tr>
<tr>
<td>ZMR *</td>
<td>Reset Font Size</td>
<td>Ctrl + 0</td>
<td>Reset the size of the font in all the windows to its default value</td>
</tr>
</tbody>
</table>